New Methods for Computing a Saddle-Node Bifurcation Point for Voltage Stability Analysis

Jin Lu
Odyssey Research Associates, Inc.
Ithaca, NY 14850
email: ljjin@oracorp.com

Chih-Wen Liu, James S. Thorp
School of Electrical Engineering
Cornell University
Ithaca, NY 14853

Abstract — This paper proposes methods for calculating saddle-node bifurcation points of power system power flow equations. The first method calculates a saddle-node bifurcation point along a given ray in the parameter space of power flow equations. By exploiting the special structure of power flow equations, the method calculates a saddle-node bifurcation point along a given ray as a solution to a constrained optimization problem. The constrained optimization can be solved efficiently with standard optimization methods. The second method calculates a locally closest saddle-node bifurcation point with respect to the operating point. This method uses an iterative process of computing a saddle-node bifurcation point along a ray, and then updating the direction of the ray for calculating a closer saddle-node bifurcation point. The method is a quasi-Newton method that updates the direction of a ray based on the first-order derivatives and the approximations to the second-order derivatives of the distance between saddle-node bifurcation points and the operating point. The paper compares the proposed methods with other methods on two test examples.

Keywords — power flow equations, saddle-node bifurcation point, constrained optimization, quasi-Newton methods

1 Introduction

A saddle-node bifurcation point of the power flow equations of a power system can provide information regarding the margin of static voltage stability at the current operating point of the power system. The information is used in voltage stability control. Over the last few years, we have seen several methods for computing saddle-node bifurcation points of power flow equations [2, 1, 3, 4, 5, 6]. For example, the continuation methods and the direct methods have been used extensively to compute saddle-node bifurcation points. While these methods provide general-purpose algorithms for computing saddle-node bifurcation points of power flow equations, the efficiency of these methods can be greatly improved if the special structures of the power system models are taken into account [5].

In this paper, we consider the power flow equations where the parameters appear linearly, namely,

\[ f(x, \lambda) = g(x) - H(x)\lambda = 0 \]  

where \( x \in \mathbb{R}^n \) is the system state vector, \( \lambda \in \mathbb{R}^p \) \((p \leq n)\) is a parameter vector, \( f(\cdot, \lambda) \in \mathbb{R}^{n+p} \rightarrow \mathbb{R}^n \), \( g(\cdot) \in \mathbb{R}^n \rightarrow \mathbb{R}^n \) and \( H(\cdot) \in \mathbb{R}^{n+p} \rightarrow \mathbb{R}^{n+p} \) are vector (or matrix) functions. Most power flow equations for static voltage stability analysis fall into this form. For example, the power flow equations where \( \lambda \) is a parameter vector of real and reactive load powers are in this form with \( h(x) \) a constant matrix.

A saddle-node bifurcation point of the power flow equation (1) is defined as \( (x, \lambda) \) that satisfy

\[ f(x, \lambda) = 0 \]  

\[ D_x f(x, \lambda) \]  

where \( D_x f(x, \lambda) \) is the Jacobian of \( f(x, \lambda) \) with respect to \( x \). We should point out that a saddle-node bifurcation point defined here is sometimes referred to as maximum loading point, critical loading conditions, voltage stability limit, etc. A direct method computes a saddle-node bifurcation point \( (x, \lambda) \) by solving the following nonlinear equations.

\[ f(x, \lambda) = 0 \]  

\[ D_x f(x, \lambda) v = 0 \text{ (or } w D_x f(x, \lambda) = 0 \text{)} \]  

where \( v \) (\( w \)) is the right (left) eigenvector associated with the zero eigenvalue of \( D_x f(x, \lambda) \). In general, we must solve a set of \( 2n + 1 \) nonlinear equations when we use a direct method.

In this paper, we will exploit the structure of the model (1) and propose new methods for computing a saddle-node bifurcation point. First, we consider the problem of computing a saddle-node bifurcation point along a specified ray — a one-dimension subspace — in the parameter space. This problem arises in the power systems where the direction of the parametric changes is assumed and it is the limit of the changes along the direction that is of interest. We propose a new method to compute a saddle-node bifurcation point along the specified ray by maximizing an objective function subject to the constraints of \( n - 1 \) nonlinear equations, instead of solving \( 2n + 1 \) nonlinear equations as a direct method does. The constrained maximization can be solved efficiently using standard optimization methods. We will compare the new method with direct methods.

References

Second, we consider the problem of determining a locally closest saddle-node bifurcation point with respect to the operating point. Roughly speaking, there are two approaches to this problem. One approach uses an iterative process of computing a saddle-node bifurcation point along a ray, and then updating the direction of the ray for calculating a closer saddle-node bifurcation point [1]. The intermediate results during the iterative process are all saddle-node bifurcation points, giving a clear geometric picture of the hypersurface \( \sum \) of the saddle-node bifurcation points. As a result, this process almost always converges to a locally closest saddle-node bifurcation point. However, this approach often updates a ray direction based on the first-order derivatives of the distance between saddle-node bifurcation points and the operating point with respect to the ray direction. Because only the first-order derivatives are used, the convergence rate of this approach can be slow when the dimension of the parameter space is large [9]. The other approach formulates the problem of determining a locally closest saddle-node bifurcation point as a constrained minimization problem where the objective function is the distance between a saddle-node bifurcation point and the operating point, and the constraints are the equations (4) [8]. Newton-type methods are used to solve the minimization, therefore the convergence rate of the computation of this approach can be faster than that of the first approach. However, since the geometric picture of the hypersurface \( \sum \) of the saddle-node bifurcation points is lost in the process of minimization, this type of direct methods may result in points that satisfy the constraints (4), yet are not physically meaningful, if the initial conditions are not properly chosen [1]. 

In light of the pros and cons of the two approaches, we propose a quasi-Newton method based on the first approach. The method updates a ray direction based on the approximation to the second-order derivatives as well as the first-order derivatives of the distance between saddle-node bifurcation points and the operating point with respect to the ray direction.

Finally, we will compare numerically our methods with some existing methods on two test examples.

2 Preliminaries

In this section, we give definitions and results that are useful in this paper.

Definition 1: \( p \) vector functions \( \varepsilon_i(z) \in \mathbb{R}^n \), \( i = 1, \cdots, p \) are linearly independent, if there exists no constant and non-zero vector \( n = [n_1, \cdots, n_p]^T \) such that
\[ \varepsilon_1(z), \cdots, \varepsilon_p(z) n = 0, \quad \forall z \] (5)

Definition 2: An \( n \times p \) (\( p \leq n \)) matrix function \( H(z) \) is a full-rank matrix function, if the \( p \) columns are linearly independent.

Definition 3: A matrix function \( H(z) \) is a \( C^2 \) matrix function if all of its components are \( C^2 \) functions. Assume that in (1), \( \dot{g}(z) \) is a \( C^2 \) vector function, \( H(z) \) is a full-rank \( C^2 \) matrix function. We know from linear algebra [10] that there exists a full-rank \( C^2 \) matrix function \( T(z) = [t_{ij}(z)]_{n \times n} \) such that
\[ T(z)H(z) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \] (6)

where \( I \) is a \( p \times p \) identity matrix, 0 is an \((n - p) \times p\) zero matrix. Pre-multiplying (1) with \( T(z) \), we have
\[ T(z)g(z) = \begin{bmatrix} I \\ 0 \end{bmatrix} \lambda = \dot{g}(z) - \begin{bmatrix} \lambda \\ 0 \end{bmatrix} = 0 \] (7)

where \( \dot{g}(z) = T(z)g(z) \).

In practice, \( T(z) \) can be easily obtained by taking into account the structure of \( H(z) \). For example, \( H(z) \) often takes a "diagonal" form:
\[ H(z) = \begin{bmatrix} \bar{H}(z) \\ 0 \end{bmatrix}, \quad \bar{H}(z) = \text{diag}([h_{11}(z), \cdots, h_{pp}(z)]) \] (8)

where \( h_{ii}(z) \) is a function that only involves the states associated with the corresponding bus (e.g., the real and reactive powers at the \( i \)th bus only depend on the voltage at the \( i \)th bus [11]). As a result, \( T(z) \) that satisfies (6) also has a very simple diagonal form
\[ T(z) = \begin{bmatrix} H^{-1}(z) & 0 \\ 0 & I \end{bmatrix} \] (9)

The nonlinear function \( g(z) \) is usually a sparse vector function — each component depends only on a few states. If \( T(z) \) is in the form (9), \( \dot{g}(z) = T(z)g(z) \) is also a sparse vector function.

In this paper, we assume that \( T(z) \) is a full-rank matrix for all \((x, \lambda)\)'s that satisfy (7).

Result 1: If \( T(z) \) is a full-rank matrix for all \((x, \lambda)\)'s that satisfy (7), then the saddle-node bifurcation points of (7) are also the saddle-node bifurcation points of (1).

Proof: See Appendix.

If \( T(z) \) is in the form (9), then it is a full rank matrix if \( h_{ii}(z) \neq 0 \) at any solutions to (7).

Because of Result 1 and the simple form of (7), we choose to study in the following the saddle-node bifurcation points of system (7).

Let \( \dot{g}(z) \) be partitioned as
\[ \dot{g}(z) = \begin{bmatrix} \dot{g}_1(z) \\ \dot{g}_2(z) \end{bmatrix} \] (10)

where \( \dot{g}_1(z) \) is the sub-vector composed of the first \( p \) elements of \( \dot{g}(z) \) and \( \dot{g}_2(z) \) is the sub-vector composed of the rest of the elements of \( \dot{g}(z) \). With this partition, (7) can be written as
\[ \begin{align*} \dot{g}_1(z) &= \lambda \\ \dot{g}_2(z) &= 0 \end{align*} \] (11)

In the next two sections, we will propose methods of computing a saddle-node bifurcation point along a given ray and a locally closest saddle-node bifurcation point of the nonlinear equations (11).

3 A Saddle-Node Bifurcation Point Along a Ray

Consider a ray in the parameter space of \( \lambda \) that starts from \( \lambda_0 \), the parameter at the operating point,
\[ \lambda = \lambda_0 + sn_0 \] (12)
where \( n_0 \) is a specified unit \( p \)-dimensional vector that defines the orientation of the ray, and \( s \) is a positive real number.

Substituting (12) into (11), we have

\[
\dot{g}_t(x) - \lambda_0 = s n_0 \tag{13}
\]

\[
\dot{g}_t(x) = 0 \tag{14}
\]

From linear algebra, there exists a full-rank, \((p-1) \times p\) matrix \( A(n_0) \) such that

\[
A(n_0) n_0 = 0 \tag{15}
\]

Matrix \( A(n_0) \) is generally a linear function of \( n_0 \), therefore we use the notation \( A(n_0) \). In fact, if \( n_{0i} \neq 0 \) \( \forall i \), where \( n_{0i} \) is the \( i \)th element of \( n_0 \), the following full-rank \( A(n_0) \) satisfies (15)

\[
A(n_0) = \begin{bmatrix}
    n_{02} & -n_{01} & 0 & 0 & 0 \\
    0 & n_{03} & -n_{02} & 0 & 0 \\
    \vdots & \vdots & \vdots & \vdots & \vdots \\
    0 & 0 & 0 & \cdots & -n_{0p} - n_{0(p-1)}
\end{bmatrix} \tag{16}
\]

Define a \( p \times p \) matrix as

\[
U = \begin{bmatrix}
    n_0^T \\
    A(n_0)
\end{bmatrix} \tag{17}
\]

It is easy to verify that \( U \) is a nonsingular matrix. Premultiplying the both side of (13) with \( U \), we have

\[
n_0^T (\dot{g}_t(x) - \lambda_0) = s \tag{18}
\]

\[
A(n_0) (\dot{g}_t(x) - \lambda_0) = 0 \tag{19}
\]

If \( \dot{g}_t(x) \) is a sparse vector function (each element depends only on a few states), then \( A(n_0) \dot{g}_t(x) \) is also a sparse vector function, where \( A(n_0) \) is in the form (16).

A saddle-node bifurcation point along the ray defined by (12) is a point \((x, \lambda_0 + s n_0)\) where \( x \) satisfy (14) and (19), and \( s \), as a function of \( x \) defined by (18), is "stretched" to its maximum. This suggests that we find a saddle-node bifurcation point by solving the following optimization problem,

\[
\max_s \max_s s = \max_n n_0^T (\dot{g}_t(x) - \lambda_0) \tag{20}
\]

subject to

\[
A(n_0)(\dot{g}_t(x) - \lambda_0) = 0 \tag{21}
\]

This optimization problem has \((n-1)\) constraints.

We note that if (19) is satisfied, then \( s \) can be expressed as

\[
s = (g_t(x) - \lambda_0)/n_{0i}, \quad \text{for} \quad n_{0i} \neq 0 \tag{22}
\]

where \( g_t(x) \) and \( \lambda_0 \) are the \( i \)th components of \( g_t(x) \) and \( \lambda_0 \), respectively. Replacing the objective function (20) with (22) might simplify the expression of (20).

Assume that a solution to the optimization (20) and (21) is \( s^* \). Calculate \( s^* \) by substituting \( s^* \) into (18). We can show that \((s^*, x^*)\) is a saddle-node bifurcation point of equations (13) and (14). First, it is obvious that \((s^*, x^*)\) satisfies (13) and (14) (or equivalently (18), (19) and (14)). Second, according to the theory of constrained optimization [9], since \( s^* \) is a local maximum point, there exists a Lagrange multiplier \( \alpha^* \) such that \((\alpha^*, s^*)\) satisfies

\[
D_x L(x^*, \alpha^*) = 0 \tag{23}
\]

\[
A(n_0)(\dot{g}_t(x^*) - \lambda_0) = 0 \tag{24}
\]

\[
\dot{g}_t(x^*) = 0 \tag{25}
\]

where \( L(x, \alpha) \) is the Lagrangian function

\[
L(x, \alpha) = n_0^T (\dot{g}_t(x) - \lambda_0) - \alpha_1 A(n_0)(\dot{g}_t(x) - \lambda_0) - \alpha_2 \dot{g}_t(x) \tag{26}
\]

with \([\alpha_1, \alpha_2]\) a proper partition of \( \alpha \). Substituting (26) into (23), we have

\[
[1, -\alpha_1^*, -\alpha_2^*] D_x \left[ \begin{array}{c}
    n_0^T (\dot{g}_t(x^*) - \lambda_0) \\
    A(n_0)(\dot{g}_t(x^*) - \lambda_0) \\
    \dot{g}_t(x^*)
\end{array} \right] = 0 \tag{27}
\]

where \( w \) is a row vector

\[
w = [1, -\alpha_1^*, -\alpha_2^*] A(n_0) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{28}
\]

Equation (27) is exactly the condition of a saddle-node bifurcation point on the Jacobian for system defined by (13) and (14); the row vector \( w \) is a left eigenvector associated with the zero eigenvalue of the Jacobian.

We can use any standard nonlinear programming algorithms to solve the optimization problem (20) and (21) [9]. As shown by the discussion in the previous paragraph, calculating a saddle-node bifurcation point by direct methods is equivalent to calculating a saddle-node bifurcation point by our method if the Lagrangian function algorithm is used to solve the optimization problem (20) and (21). However, the optimization problem (20) and (21) may be solved more efficiently by other methods [9].

Forming the problem of calculating a saddle-node bifurcation point as a constrained optimization problem makes it easy to take into account of constraints such as the limits on the state vector [5]. (Of course, the solution may not be a saddle-node bifurcation point, because additional constraints must be satisfied by the solution.) Another minor advantage of calculating a saddle-node bifurcation point as a solution to a constrained optimization problem is that it guarantees the solution \( s \) is positive (as a result of maximization), while the \( s \) obtained by a direct method can be negative if initial conditions are not properly chosen (see Example 1).

The methods of calculating saddle-node bifurcation points in [5][7] can be considered as special cases of our method, where the \( \lambda \) is a vector of reactive powers.

This section is ended with a simple example to illustrate our method.

**Example 1**: Consider the simple one-machine-infinite-bus load flow equations given in [1]:

\[
-4 V \sin \alpha - P = 0 \\
-4 V^2 + 4 V \cos \alpha - Q = 0 \tag{29}
\]
The parameter vector is $\lambda = [P, Q]^T$. The load flow equations are already in the form of (11). Assume the operating equilibrium is given by $x_0 = [V_1, V_2]^T = [-0.138, 0.909]^T$, and the current load powers are $\lambda_0 = [P_0, Q_0]^T = [0.5, 0.3]^T$.

We want to find a saddle-node bifurcation point along a ray with unit vector direction $n_0 = [n_{01}, n_{02}]^T$.

$$\lambda = \lambda_0 + sn_0, \quad s \geq 0 \quad (30)$$

Substituting (30) into (29) and after some manipulation, we have

$$-\frac{1}{n_{01}}(4V\sin \alpha + P_0) = s$$

$$4V^2 - 4V\cos \alpha - \frac{n_{02}}{n_{01}}4V\sin \alpha + Q_0 - \frac{n_{02}}{n_{01}}P_0 = 0$$

Now, let $x = V\cos \alpha$ and $y = V\sin \alpha$, we have from (31)

$$\lambda = \lambda_0 + sn_0 \quad (32)$$

where $s^2 = 1/4 + (\frac{n_{02}}{2n_{01}})^2 + \frac{n_{02}}{4n_{01}}P_0 - Q_0/4.$

The saddle-node bifurcation point can be found by solving this optimization

$$\max_s s = -\frac{P_0}{n_{01}} - \frac{4}{n_{01}}\min_y y$$

subject to

$$(x - 0.5)^2 + (y - \frac{n_{02}}{2n_{01}})^2 = s^2$$

The solution to this optimization is

$$y^* = \frac{n_{02}}{2n_{01}} - \frac{1}{2}\sqrt{1 + (\frac{n_{02}}{n_{01}})^2 + \frac{n_{02}}{n_{01}}P_0 - Q_0} \quad (35)$$

$$s^* = -\frac{1}{n_{01}}(P_0 + 4y^*) \quad (36)$$

Using $n_0 = [n_{01}, n_{02}]^T = [0.7071, 0.7071]^T$ and $[P_0, Q_0]^T = [0.5, 0.3]^T$, we have

$$s^* = 0.6567 \quad (37)$$

and the corresponding parameter vector and the state vector are $\lambda^* = [0.9644, 0.7644]^T$ and $[\alpha^*, V^*] = [-0.4501, 0.3553]$, respectively.

It is easy to show that for $n_0 = [0.7071, 0.7071]^T$, $s = -7.7308$ and $[\alpha^*, V^*] = [-0.4501, 1.3385]$ satisfy equations (4) with the function $f$ replaced by (29). Therefore, a direct method applied to this example may end up with a negative $s$.

### 4 A Locally Closest Saddle-Node Bifurcation Point

It is of great interest to know the distance between the current operating point and a locally closest saddle-node bifurcation point [1]. A commonly used approach to searching for a locally closest saddle-node bifurcation point consists of the following steps:

1. Choose an initial ray direction $n_0$, let the iteration number $i = 0$;
2. Calculate a saddle-node bifurcation point along the given ray defined by $n_i$;
3. Calculate a new ray direction $n_{i+1}$, based on the information gathered at the $i$th iteration, so that a closer saddle-node bifurcation point can be found.
4. Let the iteration number $i = i + 1$, and repeat step (2) and (3) until convergence of $n_i$.

The saddle-node bifurcation point along a given ray in Step (2) can be calculated using the method presented in the previous section or other methods. In this section, we will focus on the solution to Step (3).

For a given unit vector $n$, let $s(n)$ denote the distance between $\lambda_0$ and a saddle-node bifurcation point $\lambda(n)$ along $n$, that is, $\lambda(n) = \lambda_0 + s(n)n$.

The problem of determining a locally closest saddle-node bifurcation point can be formulated as an optimization problem,

$$\min_n s(n) \quad (38)$$

subject to

$$n^Tn - 1 = 0 \quad (39)$$

Let the current ray direction be $n_i$. We use a quasi-Newton approach to update $n_i$ to obtain a new ray direction $n_{i+1}$.

Such an approach has two elements:

(a) use a quadratic function $s'(n)$ to approximate $s(n)$ in the neighborhood of $n_i$;

(b) find a new ray direction $n_{i+1}$ based on the quadratic function $s'(n)$.

Some methods use the normal vector to the hypersurface $\sum$ of saddle-node bifurcation points at $\lambda(n_i)$ as the new ray direction $n_{i+1}$. In section 4.2, we will show that using the normal vector as $n_{i+1}$ is equivalent to using the first-order derivative of $s(n)$ at $n_i$ as $n_{i+1}$. We can compare these type of methods with steepest decent methods; they may result in slow convergence, because only the first-order derivatives are used [9].

We will discuss Steps (a) and (b) in the following.

#### 4.1 Approximation of $s(n)$ with a Quadratic Function

Consider the following quadratic function obtained by neglecting terms whose orders are higher than two in the Taylor expansion of $s(n)$ around $n_i'$,

$$s'(n) = (n - n_i')^T \frac{ds(n)}{dn} |_{n_i'} + \frac{1}{2}(n - n_i')^T \frac{d^2s(n)}{dn^2} |_{n_i'}(n - n_i') \quad (40)$$

where $\frac{ds(n)}{dn}$ and $\frac{d^2s(n)}{dn^2}$ are the first-order derivatives and Hessian of $s(n)$ at $n_i'$, respectively.

The following proposition gives a result on the calculation of $\frac{ds(n)}{dn}$.

**Proposition 1:**

Let $w = [w_1, w_2]$ be a row vector associated with the zero eigenvalue of the Jacobian of (11) at the saddle-node bifurcation point $(x', \lambda')$, where $\lambda' = \lambda_0 + s(n')n'$, that is,

$$[w_1, w_2]\begin{bmatrix} D_x \hat{g}_1(x)|_{x'} & D_x \hat{g}_2(x)|_{x'} \end{bmatrix} = 0 \quad (41)$$

where $D_x \hat{g}_i(x)$ is the Jacobian of $\hat{g}_i(x)$, $i = 1, 2$. Then

$$\frac{ds(n)}{dn} |_{n_i'} = -s(n')n_i'^T \quad (42)$$

**Proof:** See Appendix.
The computation of \( \frac{d^2 f(n)}{dn^2} \) is rather difficult. Therefore, we will follow the way the Hessian is approximated and updated in a quasi-Newton algorithm, namely, we use a symmetric matrix to approximate it at the beginning of our iteration, and update it as we accumulate more points on the hypersurface of \( s(n) \).

Let \( \delta' = n' - n^{i-1} \) and \( \gamma' = \frac{d_s(n)}{dn} \mid_{n^i} - \frac{d_s(n)}{dn} \mid_{n^{i-1}} \). Let \( H' \) denote the approximation to \( \frac{d^2 s(n)}{dn^2} \mid_{n^i} \). We require that \( H' \) behave like the Hessian of a quadratic function: (1) \( H' = (H')^T \); (2) \( \gamma' = H' \delta' \); (3) \( H' = H' + \Delta' \), where \( \Delta' \) is an updating term.

The BFGS formula [9] is a good candidate for updating \( H' \) that satisfies the above three requirements:

\[
H' = H' + \gamma'(\gamma')^T - (\gamma' H')^{-1} \gamma'
\]

Therefore, we will use it to approximate \( \frac{d^2 s(n)}{dn^2} \mid_{n^i} \) in the following.

### 4.2 Updating \( n' \) to Obtain \( n'^{i+1} \)

Substituting (40) into (38) with \( H' \) in place of \( \frac{d^2 s(n)}{dn^2} \mid_{n^i} \), we have

\[
\min_{n} (n - n')^T \frac{d_s(n)}{dn} \mid_{n^i} + \frac{1}{2} (n - n')^T H'(n - n')
\]

subject to

\[
n^T n - 1 = 0
\]

Let \( n' \) be the solution to the optimization (43) and (44). The following proposition characterizes \( n' \).

**Proposition 2:**

\[
n' = -(H' + \alpha I)^{-1} \left( \frac{d_s(n)}{dn} \mid_{n^i} - H' n' \right)
\]

where \( \alpha \) is a scalar such that \( (H' + \alpha I) \) is positive definite and \( ||n'|| = 1 \).

**Proof:** See the corollary on Page 102 of [9].

\( \delta' \) in (45) can be obtained by substituting (45) into ||\( n' || = 1, \) and solving the resulting equation. We know that the symmetric matrix \( H' \) can be decomposed as the Schur decomposition \( H' = Q \Omega Q^T \), where \( Q = [q_1, \cdots, q_n] \) is an orthonormal matrix and \( \Omega = \text{diag}(\omega_1, \cdots, \omega_n) \) is a diagonal matrix with eigenvalues \( \omega_1 \geq \cdots \geq \omega_n \). Substituting \( H' = Q \Omega Q^T \) into (45), we have

\[
n' = \sum_{i=1}^{n} \beta_i q_i
\]

where \( \beta_i = -c_i / (\omega_i + \alpha) \), \( c_i = q_i^T \frac{d_s(n)}{dn} \mid_{n^i} - H'n' \). Substituting (46) into ||\( n' || = 1, \) we have

\[
\sum_{i=1}^{n} \beta_i^2 = 1
\]

It is easy to show that the LHS of (47) increases from 0 to \( \infty \) and \( (H' + \alpha I) \) is positive definite, as \( \delta' \) decreases from \( \infty \) to \( -\infty \). Therefore, there exists an \( \delta' \in (-\infty, \infty) \) such that ||\( n' || = 1 \) and \( (H' + \alpha I) \) is positive definite. Such a solution can be calculated efficiently with the method in [12].

\( n' \) gives the direction to a locally closest point on \( s(n) \). Since \( s'(n) \) is an approximation to \( s(n) \) around \( n' \), \( n' \) may not give the direction to a locally closest point on \( s(n) \). Therefore, instead of using \( \delta' \) as \( n'^{i+1} \), we may consider the following set of ray directions,

\[
n(a) = -c(H' + \alpha I)^{-1} \left( \frac{d_s(n)}{dn} \mid_{n^i} - H' n' \right)
\]

where \( c \) is chosen such that \( n(a) \) is a unit vector. We can use a line search to find an \( \alpha' \) such that

\[
s(n(a')) = \min_{\alpha} s(n(\alpha))
\]

where the function \( n(\alpha) \) is defined by (48). Then, let \( n^{i+1} = n(\alpha') \). \( \alpha' \) can be used as an initial guess in the line search. Many good line search methods can be used to carry out the line search [9].

We mentioned that some methods use the normal vector to the hypersurface \( \sum_{i} \) of saddle-node bifurcation points at \( \lambda(n) \) as the new ray direction \( n^{i+1} \). Since the normal vector is proportional to \( \frac{d_s(n)}{dn} \mid_{n^i} = -\frac{n(n)}{w_i} \) [1], this type of methods can be considered as a special case of our method where \( H' = 0 \).

### 4.3 Algorithm

The following algorithm summarizes our approach to computing a closest saddle-node bifurcation point.

**Algorithm 1:**

1. Choose an initial unit vector \( n^i \), and let \( H' = I \) and \( i = 0 \);
2. Compute \( s(n^i) \), \( \lambda^i = \lambda_0 + s(n^i) \), and let \( w_i \) that satisfies (41);
3. Let \( \frac{d_s(n)}{dn} \mid_{n^i} = -\frac{n(n)}{w_i} \), and do the following:
   - (3.1) Calculate \( H' \) using formula (42);
   - (3.2) Find \( \alpha' \) such that \( s(n(a')) = \min_{\alpha} s(n(\alpha)) \), where \( n(\alpha) \) satisfies (48) and let
     \[
n^{i+1} = -c(H' + \alpha I)^{-1} \left( \frac{d_s(n)}{dn} \mid_{n^i} - H' n' \right)
\]
   where \( c \) is chosen such that \( n^{i+1} \) is a unit vector.
4. If \( ||n^{i+1} - n^i|| \leq \varepsilon \), where \( \varepsilon \) is a pre-specified, small number, then stop; otherwise, let \( i = i + 1 \), repeat Step (2)-(3).

Step (2) of Algorithm 1 can be handled with the method presented in the previous section or other methods. If a Lagrangian function algorithm is used to solve the optimization (20) and (21), then \( w_i \) is obtained as part of the result (see the discussion before Example 1). Some other methods also obtain \( w_i \) as a by-product of calculating a saddle-node bifurcation point [1].

As previously mentioned, many standard line search techniques in quasi-Newton optimization methods can be used to carry out the line search in Step (3.2). Each iteration of the line search needs to calculate a saddle-node bifurcation point along a given ray. Here the exact line search is not necessary. A good line search algorithm can usually find a satisfactory \( \alpha \) in a few iterations with a good initial guess (e.g. \( \alpha' \) in (45)).
At the first several iterations of Algorithm 1, $H^i$ may not be a good approximation to the Hessian of $s(n)$. Therefore, at the first several iterations, we update $H^i$ as we gather more information, but only use the first-order derivatives $\frac{ds(n)}{dn}$, for updating $n^i$, namely, let $n^{i+1} = c \frac{ds(n)}{dn}$. After the first several iterations, we consider $H^i$ to be a good approximation to the Hessian, and use formula (50) for updating $n^{i+1}$. Therefore, Step (3.2) can be modified as

$$
(3.2') \text{If } i \leq M \ (M \text{ is a pre-specified integer), then } n^{i+1} = c \frac{ds(n)}{dn}; \text{ otherwise, find } \alpha' \text{ such that } s(n(\alpha')) = \min_{n} s(n(\alpha)), \text{ where } n(\alpha) \text{ satisfies (49), and let}
$$

$$
n^{i+1} = -c(H^i + \alpha'T)^{-1}(\frac{ds(n)}{dn}|_{n^i} - H'n^i) \quad (51)
$$

where $c$ is chosen such that $n^{i+1}$ is a unit vector.

The prespecified integer $M$ depends on the dimension of the power flow equations.

In Step (3.2) (or Step (3.2')), we may need the Schur decomposition of $H'$ to find $\alpha'$ in (45) as an initial guess for the line search. Also, we can use the Schur decomposition of $H'$ to make the computation of formula (50) more efficient.

## 5 Examples

In this section, we use Algorithm 1 to calculate locally closest saddle-node bifurcation points for two test examples.

First, consider the simple load flow equations (29) in Example 1. The result of applying Algorithm 1 to the equations is shown in Table 1. The $s^i$ in the table is obtained in Step (3.2) of Algorithm 1, and $\alpha^i$ is associated with $n^i$, the optimal ray direction with respect to the quadratic approximation $s^i(n)$. In this example, $\alpha'$ and $\alpha^i$ are so close, showing that $s^i(n)$ is a good approximation to $s(n)$. For this example, we could simply use $n^i$ as $n^{i+1}$, without searching for a proper $\alpha$ used in the formula (50). In general, this may not be the case. The convergence of $n^i$ is reached at the iteration 2.

The parameter vector and the state vector at the closest saddle-node bifurcation point are $\rho = [0.5, 0.3]^T + 0.611 x [0.331, 0.944]^T = [0.703, 0.877]^T$, and $x = [-0.338, 0.530]^T$, respectively. Paper [1] also calculated the closest saddle-node bifurcation point for this two bus system, using a method that is similar to our method but updates the ray direction based on the normal vector to the hypersurface $\sum_{i} n_i$ of saddle-node bifurcation points (this is equivalent to updating the ray direction based on the derivatives of $s(n)$). Our result is the same as that in paper [1], but the convergence rate of $n^i$ here is faster compared with the method in paper [1].

Second, consider the five bus system given in [2] (see Fig. 2). The system has two generators at buses 1 and 3, and three load buses 2, 4 and 5. The line parameters, bus data, and the base case power flow results can be found in [2]. The real and reactive powers consumed by the loads are chosen as parameter vector $\lambda = [P_2, Q_2, P_4, Q_4, P_5, Q_5]^T$. The state vector $x = [\theta_1, \theta_2, \delta_1, \delta_2, V_2, V_4, V_5]^T$. The load flow equations are in the form of (11). The iteration history of Algorithm 1 applied to this example is listed in Table 2. The convergence of the algorithm is reached at iteration 4. In the first 3 iterations, we only use $\frac{ds(n)}{dn}$, for obtaining $n^{i+1}$ (the $M$ in the Step (3.2') of Algorithm 1 is 3). The parameter and the state at the closest saddle-node bifurcation point are $\lambda^* = 0.8938; \ x^* = [0.1485, 0.6008, 1.1659, 1.0495, 0.7491, 0.5320]^T$.

### Table 1: Results of Algorithm 1 applied to the 2 bus system

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n^i$</th>
<th>$s^i$</th>
<th>$\frac{ds(n)}{dn}$</th>
<th>$H^i$</th>
<th>$\alpha^i$</th>
<th>$\hat{\alpha}^i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$[0.585, 0.811]^T$</td>
<td>0.6304</td>
<td>$[-0.2571, -0.5919]^T$</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
<td>$-0.0549$</td>
<td>0.3115</td>
</tr>
<tr>
<td>1</td>
<td>$[0.398, 0.917]^T$</td>
<td>0.6123</td>
<td>$[-0.2137, -0.5747]^T$</td>
<td></td>
<td>0.3115</td>
<td>0.7088</td>
</tr>
<tr>
<td>2</td>
<td>$[0.33, 0.944]^T$</td>
<td>0.6112</td>
<td>$[-0.2023, -0.5676]^T$</td>
<td></td>
<td>$-0.0476$</td>
<td>0.3066</td>
</tr>
</tbody>
</table>

### Table 2: Results of Algorithm 1 applied to the 5 bus system ($*: H^i$ is not used)

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n^i$</th>
<th>$s^i$</th>
<th>$\alpha^i$</th>
<th>$\hat{\alpha}^i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$[0.4017, 0.0063, 0.8188, -0.0373, 0.4083, 0.0085]^T$</td>
<td>1.5976</td>
<td>0.9312</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>$[0.0380, 0.0063, 0.7608, 0.6153, 0.0985, 0.1773]^T$</td>
<td>0.8965</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$[0.0103, 0.0011, 0.5426, 0.8247, 0.0563, 0.1490]^T$</td>
<td>0.8940</td>
<td>0.892</td>
<td>0.8942</td>
</tr>
<tr>
<td>3</td>
<td>$[0.0095, 0.0009, 0.5213, 0.8387, 0.0548, 0.1473]^T$</td>
<td>0.8938</td>
<td>0.894</td>
<td>0.8937</td>
</tr>
</tbody>
</table>
6 Conclusion

This paper has proposed two methods for calculating saddle-node bifurcation points of power flow equations. The first method exploited the special structure of power flow equations and formulated the problem of calculating a saddle-node bifurcation point along a given ray as a constrained optimization problem that can be solved efficiently with standard optimization methods. The second method calculated a locally closest saddle-node bifurcation point using an iterative process of computing a saddle-node bifurcation point along a ray and updating the direction of the ray for calculating a closer saddle-node bifurcation point. The method updates the direction of the ray based on the approximations to the second-order derivatives as well as the first-order derivatives of the distance between saddle-node bifurcation points and the operating point. Comparison of our methods with other methods on two test examples showed the effectiveness of our methods. Further tests of our methods on large power systems are needed to verify the efficiency and robustness of our methods.

References


Appendix

Proof of Result 1:

We will prove Result 1 by showing that under the condition of Result 1, (a) $f(x, \lambda)$ satisfies equation (1) if $f(x, \lambda)$ satisfies equation (7); (b) The Jacobians of (1) and (7) have the same rank at any point $(x, \lambda)$ that satisfies (7).

The proof of (a) is obvious, thus omitted. We will give the proof of (b).

The Jacobian of equation (1) is

$$\nabla g(x) - \sum_{i=1}^{p} \nabla h_i(x) \lambda_i \tag{52}\$$

where $\nabla g(x)$ is the Jacobian of a vector function $g(x)$, $\nabla h_i(x)$ is the Jacobian of $h_i(x)$, the ith column of $H(x)$.

From $T(x)H(x) = \begin{bmatrix} I \\ 0 \end{bmatrix}$, we have

$$\nabla h_i(x) = - T^{-1}(x) \begin{bmatrix} \frac{dT(x)}{dx} h_i(x) \\ \frac{dT(x)}{dx} h_i(x) \end{bmatrix} \tag{53}$$

By substituting (53) into (52), the Jacobian of (1) becomes

$$\nabla g(x) + T^{-1}(x) \begin{bmatrix} \frac{dT(x)}{dx} h_i(x) \\ \frac{dT(x)}{dx} h_i(x) \end{bmatrix} \frac{dT(x)}{dx} H(x) \lambda \tag{54}$$

Since $T(x)$ is of full rank at any point $(x, \lambda)$ that satisfies (7), and the Jacobian of (7) is

$$T(x)D_g(x) + \begin{bmatrix} \frac{dT(x)}{dx} g(x) \\ \frac{dT(x)}{dx} g(x) \end{bmatrix} \tag{55}$$

the Jacobians of (1) and (7) have the same rank.

Proof of Proposition 1:

By an argument similar to the argument that leads to the constrained optimization (20) and (21), we have

$$s(n) = \max_{s_{+,-}} s \tag{56}$$

subject to

$$\begin{bmatrix} \bar{g}_1(x) \\ \bar{g}_2(x) \end{bmatrix} = 0 \tag{57}$$

where $\lambda(n) = \lambda_0 + s(n)$. The Lagrangian function of the above optimization is defined as

$$L(x, \sigma, \alpha) = s - \sigma_1 \bar{g}_1(x) - \sigma_2 \bar{g}_2(x) \tag{58}$$

where $\alpha = [\alpha_1, \alpha_2]$ is the Lagrange multiplier.

By the theory of constrained optimization (see Page 198 of [9]),

$$\frac{\partial L(x, \sigma, \alpha)}{\partial \lambda} = 0 \tag{59}$$

From equations (59) and (60), we have

$$\frac{d\lambda(n)}{dn} = \sigma_1^T T = - \frac{w_0^T}{w_1^T} \tag{61}$$

where $w_1$ is any non-zero vector that satisfies (41) for some $w_2$.

By the chain rule, the derivative of $s$ with respect to $n$ can be written as

$$\frac{ds(n)}{dn} = \frac{d\lambda(n)}{dn} \frac{d\lambda(n)}{dn} = - \frac{w_1^T}{w_1^T} \tag{62}$$

□
Jin Lu received the B.S. and M.S. degrees in electrical engineering from Nanjing Institute of Technology, Nanjing, China in 1982 and 1983 respectively. He received the Ph.D. in electrical engineering from Cornell University in 1990. He is currently an electrical engineer at the ORA Corporation, a computer software firm. His research interests are in the areas of control and optimization of large scale systems with an emphasis on power systems.

Chih-Wen Liu received the B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1987. He received M.S. in 1992 and the Ph.D. in 1994 from Cornell University in Electrical Engineering. His research interests include analysis of power system dynamics and applications of real-time phasor measurements to power system protection and control.

James S. Thorp (S'58-M'63-SM'80-F'89) received the B.E.E, M.S., and Ph.D. degrees from Cornell University, Ithaca, NY. He joined the faculty at Cornell in 1962 where he is currently a Professor of Electrical Engineering. In 1976 he was a Faculty Intern at the American Electric Power Service Corporation. He was an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1985 to 1987. In 1988 he was an Overseas Fellow at Churchill College, Cambridge, England. He is a member of the IEEE Power System Relaying Committee, CIGRE, Eta Kappa Nu, Tau Beta Pi, Sigma Xi.
Discussion

Claudio A. Cañizares (University of Waterloo): This interesting and thorough paper presents a new perspective on the computation of saddle-node bifurcations through the use of optimization techniques. Although the authors have done a remarkable job in trying to address all possible concerns with regard to the application of the proposed techniques, this discusser would appreciate their comments on the following two additional points:

1. In the proposed optimization problems for the computation of saddle-node bifurcations, including the closest one to a given operating point, the corresponding objective and constraints functions are derived based on the specific system model (1). Most typical quasi-static system and load models used in transient stability analysis can be represented by these equations; however, one cannot always guarantee a “diagonal structure” of matrix \( H(x) \). In these cases, obtaining the proposed \( g(x) \) function could be cumbersome. Have the authors studied the computational burden of calculating \( g(x) \) for other structures of the \( H(x) \) matrix?

2. In [A], the discusser and co-author proposed different techniques to improve classical direct methods to efficiently calculate saddle-node bifurcations in large power systems. The authors mention that the optimization problem represented by equations (20) and (21) is computationally more efficient than these direct methods; however, there are no results in the paper to corroborate this assertion. Have the authors tested and, hence, believe are the most appropriate methods to medium size power systems, such as 30 to 300 bus systems?

As the authors well mention in the paper, the proposed techniques need of further testing before they can be adopted by the power system community. It would be interesting to see the results of applying these methods to large power systems, to compare their efficacy with the well tested direct and continuation methods.


R. Fischl and T. Wu (Drexel University): We would like to congratulate the authors for their achievement on developing new methods for computing a saddle-node bifurcation point for voltage stability analysis. The authors have proposed two methods in their paper: one for the computation of a saddle-node bifurcation point in a given ray direction in the parameter space, and the other for the identification of the closest saddle-node bifurcation with respect to a given operating point. We have some questions concerning the application of their methods and we would appreciate the authors’ comments:

1. Since a solution to a nonlinear optimization problem usually depends on the initial estimate of the solution, what methods did the authors use to obtain the initial estimate in order to guarantee convergence to the desired optimal solution?

2. What methods would the authors use to find \( T(x) \) in (6) for the case when \( H(x) \) does not contain a simple “diagonal” submatrix? Also could the authors comment on the computational complexity when computing \( T(x) \) in this case.

3. Could the authors comment on the applicability of their methods to small power systems, such as 3 to 10 bus systems?

Manuscript received August 16, 1994.
where subscripts 'g' and 'l' denote generation and load and \( P_g \) and \( Q_g \) are zero if 'l' is greater than 'm'.

The problem with this formulation was that the optimization process could not enforce voltages at their specified values if reactive injections were found to be within limits. In this context, it will be advantageous to know authors' experience as to how a PV bus can be handled if authors' formulations are used.

Authors have demonstrated the potential of their methods on a small and simple five bus test system. While the proposed methods are yet to be applied to large sized practical systems, it will be interesting if authors could give a comparison of the proposed methods with those proposed by Dobson et al in ref. [1] of the paper in terms of speed and CPU time.

Once again commending the authors for their important contribution to voltage stability analysis, the discussers request for their valuable comments.

Manuscript received August 22, 1994.

Jin Lu (Odyssey Research Associates, Inc. Ithaca, NY 14850): The authors would like to thank the discussors for their interest, comments and questions. We lumped the questions into several types and answer them as follows:

1. What methods would the authors use to find the \( T(z) \) in (6) for the case where \( H(z) \) does not contain a simple "diagonal" form. What is the complexity of obtaining \( T(z) \) (Prof. Canizares, Prof. Fischl and Wu)

To convert eqn.(1) to eqn.(7), we need to find the quasi-inverse of \( H(z) \) (\( T(z)H(z) = I \)). We know how to do this if \( H(z) \) is a constant matrix. In the case of a matrix function, the quasi-inverse can be found symbolically. In other words, all the standard matrix manipulation techniques can be used to find the \( T(z) \); the variable \( z \) is manipulated symbolically. In the case of a "diagonal structure" of \( H(z) \), it is easy to write \( T(z) \). If \( H(z) \) is dense and the dimension of the parameter \( \lambda \) is high, then \( T(z) \) can be very complicated. Fortunately, \( H(z) \) is a sparse matrix in most cases and the special structure of \( H(z) \) can be exploited to simplify the computing of \( T(z) \). Some commercial tools are available for symbolic computing of \( T(z) \), e.g., MAPLE. We have not, however, tested the scalability of MAPLE and exploited any special structures of \( H(z) \).

In fact, what we really need to do in converting the problem of computing a saddle-node bifurcation point along a ray to the constrained minimization (20) and (21) is the following:

(a) Substitute \( \lambda \) with \( \lambda_0 + s \bar{n}_0 \) in equ. (1), where \( n_0 \) is a ray direction, to obtain

\[ f(z) - H(z)(\lambda_0 + s \bar{n}_0) = \bar{f}(z) - sg(z) = 0 \]  

where \( \bar{f}(z) = f(z) - H(z)\lambda_0 \) and \( g(z) = H(z)n_0 \) are \( n \)-dimensional vector functions.

(b) Define an \( (n-1) x n \) matrix function \( A(z) \) as

\[ A(z) = \begin{bmatrix} g_1(z) & -g_2(z) & 0 & \cdots & 0 & 0 \\ 0 & g_2(z) & -g_3(z) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & g_{n}(z) & -g_{n-1}(z) \end{bmatrix} \]  

where \( g_i(z) \) is the \( i \)-th element of \( g(z) \).

(c) Define an \( n \times n \) matrix function \( U(z) \) as

\[ U(z) = \begin{bmatrix} \frac{1}{g^T(z)g(z)} g^T(z) \bar{f}(z) \end{bmatrix} \]  

It is easy to check that

\[ U(z)(\bar{f}(z) - sg(z)) = 0 \]  

\[ \Rightarrow \frac{1}{g^T(z)g(z)} g^T(z)\bar{f}(z) = s \]  

\[ A(z)\bar{f}(z) = 0 \]

(d) Formulate the problem of finding a saddle-node bifurcation point as

\[ \max_s \frac{1}{s} \frac{1}{g^T(z)g(z)} g^T(z)\bar{f}(z) \]  

subject to

\[ A(z)\bar{f}(z) = 0 \]

We note that if (A8) is satisfied, \( s \) can be equivalently expressed as

\[ s = \bar{f}_i(z)/g_i(z), \quad \text{for } g_i(z) \neq 0 \]

where \( \bar{f}_i(z) \) is the \( i \)-th component of \( \bar{f}(z) \). Replacing the objective function (A7) with (A9) might simplify the expression of (A7).

2. What are the advantages of formulating the problem of computing a saddle-node bifurcation point along a ray as a minimization problem (20) and (21) (Prof. Canizares)

The problem of computing a saddle-node bifurcation point along a ray is formulated as a minimization problem (20) and (21) for the following reasons:

(a) To handle constraints such as reactive limits on generator buses. A classical direct method has difficulties in incorporating constraints, as the discussor pointed out. (It is difficult to solve a set of nonlinear equa-
tions subject to some nonlinear inequalities). The con-
strained minimization problem (20) and (21) can eas-
ily incorporate constraints, especially, the inequalities,
into the problem. In fact, the author of the ref. [5]
of our paper has formulated the problem of finding a
saddle-node bifurcation point as a maximization prob-
tem to incorporate generator reactive power limits into
the problem. Of course, when such limits are added to
the constrained optimization, care must be exercised in
handling the "discontinuities" in switching from a PV
bus to a PQ bus when some generator reactive power
limits are reached. In this case, the optimization be-
comes a nonsmooth optimization.

(b) To solve the problem more efficiently than a classical
direct method. We solved the constrained minimization
(20) and (21) using a sequential quadratic programming
(SQP) algorithm. The minimization (20) and (21) can
be solved by using the method of Lagrange multipliers.
The paper showed that solving the minimization
(20) and (21) using the method of Lagrange multipliers
is equivalent to a classical direct method. Therefore,
to compare our solution method with a classical direct
method, let us compare our solution method with the
method of Lagrange multipliers. The method of La-
grange multipliers has the following drawbacks: (i) it
may converge to a solution that corresponds to a max-
imum point or a saddle point (instead of a minimum
point), because it uses only the first order condition
of an extreme point (see page 145 of ref. [9] of our
paper); (ii) generally speaking, it is not as efficient as
a sequential quadratic programming, which also guar-
antees the convergence to a local minimum point (see
page 305 of ref. [9] of our paper). Having said that,
however, we must point out that no approach is a best
approach in all situations. Our experience in solving
the minimization (20) and (21) and other optimization
problems seems to support the general view that SQP is
a more efficient method in most cases than the method
of Lagrange multipliers. However, we have not tried the
proposed method on large scale systems yet.

3 How to handle the constraints such as the lim-
its on the reactive power generations in finding
the saddle-node bifurcation points? (Dr. Srivas-
tava et al)
When the limits on the reactive power generations are
added to the optimization problem (20) and (21), the
optimization becomes a nonsmooth optimization prob-
lem. This is because the power flow equations associated
with PV buses before the limits are reached (when the
voltages are fixed) become the power flow equations
associated with PQ buses afterwards (when the reactive
powers are fixed at the limit while the voltages are ad-
justed). To be specific, when a PV bus has a reactive
power limit, the constraints that the cost function (20)
is subject to become something like,
generator bus \( i = 1, \ldots, m \):
\[
\begin{align*}
P'(V_i, \theta_i) &= P_g, \quad V = V_{base} \quad &\text{if } Q'(V_i, \theta_i) < Q_g \\
P'(V_i, \theta_i) &= P_g, \quad Q'(V_i, \theta_i) = Q_g &\text{otherwise}
\end{align*}
\] (A10)
load bus \( i = m + 1, \ldots, n \):
\[
P'(V_i, \theta_i) = P_i, \quad Q'(V_i, \theta_i) = Q_i
\] (A11)
Optimization over such "piece-wise" nonlinear func-
tions is referred to as nonsmooth optimization. The
nonsmooth optimization process is a sequence of sub-
optimization problems, each of them is a standard
smooth optimization (the regular constrained optimiza-
tion). Two consecutive sub-optimization problems are
subject to different sets of constraints. A set of bound-
aries are monitored to determine when to switch from
one set of constraints to another. In the case of reactive
power limits, the boundaries to be monitored are
\[
Q'(V_i, \theta_i) = Q_g, \quad i = 1, \ldots, m
\] (A12)
If the power flow equations for PV bus is currently used
for \( i \)th bus and boundary (A12) is reached, then the
corresponding constraints are switched to the the power
flow equations for PQ bus. If the power flow equations
for PQ bus is currently used for \( i \)th bus and boundary
(A12) is violated, then the corresponding constraints are
switched to the the power flow equations for PV bus.
The discussor is referred to ref. [9] of our paper for the
nonsmooth optimization, and ref. [5] for a method to
handle reactive power limits.

4. Could the authors comment on the applicabil-
ity of their methods to the medium (large) size
power systems? (Prof. Cañizares, Prof. Fischl and
Wu)
The first proposed method amounts to an optimal
load flow method (optimization subject to certain con-
straints). Therefore, it is applicable to medium (large)
size power systems.
The second proposed method amounts to a set of prob-
lems of finding saddle-node bifurcation points. We have
not tried the method on medium (large) size power systems.

5. How do you compare the second proposed
method with that proposed by Dobson et al? (Dr.
Srivastava et al)
The computational amount of determining a closest
saddle-node bifurcation point is dominated by the num-
ber of computing a saddle-node bifurcation point along a
given ray in the parametric space. In other words, if we
denote by \( k \) the number of computing a saddle-node
bifurcation point along a given ray, then the computa-
itional amount of determining a closest saddle-node bifurcation point is approximately O(k). (The updating of the ray direction for searching for a new saddle-node bifurcation point is computationally simple in both the proposed method and the method proposed by Dobson et al in ref. [1]). In fact, one motivation of using the second-order information about the curvature of the hypersphere of the saddle-node bifurcation points is to reduce the number of computing a saddle-node bifurcation point along a given ray. In the first test example, the number of computing a saddle-node bifurcation point along a given ray is 3 before termination; in the second test example, the number is 5 before termination. In both cases, the numbers of computing a saddle-node bifurcation point along a given ray are smaller than those of the method proposed by Dobson et al in ref. [1]. The reason is probably because the second-order approximations to the hypersphere of the saddle-node bifurcation points in both examples are quite good approximations. (see the $\alpha^1$ and $\alpha^2$ in Table 1 and 2; if they are close, it implies the second-order approximation is a good one.) Although a second-order approximation is a good approximation to the hypersphere of the saddle-node bifurcation points near a locally closest saddle-node bifurcation point, whether it is the case at the points on the hypersphere further away from a locally closest saddle-node bifurcation point remains to be studied. Testing our methods on large scale power systems will certainly provide valuable information for understanding of the problem.

6. How to choose an initial estimate in order to guarantee the convergence to the desired optimal solution? (Prof. Fischl and Wu)

We have not given much thinking to this aspect. We just chose the parameters at the operation points and start from there. It should be interesting to be able to find a good initial estimate that is close to the desired optimal solution.

Finally, thanks again for the discussors' comments and questions.

Manuscript received October 20, 1994.