Iterative Computation of Marginally Stable Trajectories

I.A. Hiskens*
Department of Electrical and Computer Engineering
University of Wisconsin - Madison
Madison, WI 53706, USA

August 4, 2003

Abstract

Stability limits place restrictions on the economic operation of power systems. Calculation of these limits is therefore very important, but has traditionally been quite difficult. This paper proposes an iterative algorithm for determining parameter values that result in marginal stability of a system. (A system is marginally stable for a particular disturbance if the post-disturbance trajectory lies on the stability boundary.) The algorithm is based on Gauss-Newton solution of a nonlinear least-squares problem. Gradient information is provided by trajectory sensitivities.

1 Introduction

In many power systems, the maximum power transfer across the network is limited by stability considerations. If the transfer level is raised too high, instability may occur for certain disturbances. To minimize this risk, operators are guided by constraints which ensure that if certain prescribed large disturbances occur, for example faults or generator tripping, then the system will continue to operate in a desirable fashion. A good knowledge of these stability constraints is very important. If the constraints are underestimated, then systems tend to be operated over-conservatively, with an associated economic penalty. However should the constraints be overestimated, then the system may well be operated in a vulnerable state without a clear understanding of the risk. Unfortunately the computation of these stability limits is difficult though.

Underlying the calculation of stability limits is the need to find a loading condition (set of parameters) at which the disturbed system is marginally stable. This set of parameters may include the generation schedule, loads, and controller setpoints, for example. Generally it is necessary to consider many disturbance scenarios, with a critical loading condition being calculated for each case. A comparison of those critical loading conditions provides the overall limiting case.

The concept is straightforward, but implementation is difficult. It is not easy to find a marginally stable system. Lyapunov-based direct methods provide a useful approach [1]. However modelling restrictions limit their application and accuracy. Otherwise, no systematic approach has been available. Hence stability limits are generally determined off-line, for a limited set of operating conditions and disturbance scenarios. The results of these studies are then used on-line to guide operators.

This paper proposes a novel approach to finding parameter values that induce marginally stable system behaviour.

It is shown in the paper that the proposed algorithm is applicable (under fairly mild assumptions) to systems that exhibit nonlinear, non-smooth behaviour. In the context of power systems, numerous devices induce such behaviour, including generators, controller limits and arbitrarily complicated protection characteristics. An appropriate model is discussed in Section 2, along with

*Research supported by the National Science Foundation through grant ECS-0114725.

some useful background system theory, and an overview of trajectory sensitivities. The proposed algorithm is described in Section 3, and examples are explored in Section 4. Conclusions are given in Section 5.

2 Background

2.1 Model

Power systems frequently exhibit a mix of continuous time dynamics, discrete-time and discrete-event dynamics, switching action and jump phenomena. A useful, non-restrictive model formulation should therefore be,

- capable of capturing this full range of continuous/discrete *hybrid system* dynamics,
- computationally efficient, and
- consistent with the development of shooting-type methods.

It is shown in [2] that these specifications can be met by a model that consists of a set of differential-algebraic equations, adapted to incorporate impulsive (state reset) action and switching of the algebraic equations. This *DA Impulsive Switched* (DAIS) model can be written in the form,

$$\dot{x} = f(x,y) + \sum_{j=1}^{r} \delta(y_{r,j}) \Big(h_j(x,y) - x \Big) \tag{1}$$

$$0 = g(x,y) \equiv g^{(0)}(x,y) + \sum_{i=1}^{s} g^{(i)}(x,y) \tag{2}$$

where

$$g^{(i)}(x,y) = \begin{cases} g^{(i-)}(x,y) & y_{s,i} < 0 \\ g^{(i+)}(x,y) & y_{s,i} > 0 \end{cases} \quad i = 1, ..., d \tag{3}$$

and

- $x \in \mathbb{R}^n$ are dynamic states, and $y \in \mathbb{R}^m$ are algebraic states;
- $\delta(.)$ is the Dirac delta. Each impulse term of the summation in (1) can be expressed in the alternative state reset form

  $$x^+ = h_j(x^-, y^-) \quad \text{when } y_{r,j} = 0 \tag{4}$$

  where the notation x^+ denotes the value of x just after the reset event, whilst x^- and y^- refer to the values of x and y just prior to the event. This form motivates a generalization to an implicit mapping $h'_j(x^+, x^-, y^-) = 0$.
- y_s, y_r are selected elements of y that trigger algebraic switching and state reset (impulsive) events respectively; y_s and y_r may share common elements.
- $f, h_j : \mathbb{R}^{n+m} \to \mathbb{R}^n$.
- $g^{(0)}, g^{(i\pm)} : \mathbb{R}^{n+m} \to \mathbb{R}^m$. Some elements of each $g^{(\cdot)}$ will usually be identically zero, but no elements of the composite g should be identically zero. Each switched $g^{(i\pm)}$ may itself have a switched form, and is defined similarly to (2). This allows a nested structure for g.

Equations (1)-(3) are a reformulation (and slight generalization) of the model proposed in [3].

It can be convenient to establish the partitions

$$x = \begin{bmatrix} \underline{x} \\ \bar{x} \\ \lambda \end{bmatrix}, \qquad f = \begin{bmatrix} \underline{f} \\ 0 \\ 0 \end{bmatrix}, \qquad h_j = \begin{bmatrix} \underline{x} \\ \bar{h}_j \\ \lambda \end{bmatrix} \tag{5}$$

where $\underline{x}$ are the continuous dynamic states, $\bar{x}$ are discrete states, and λ are parameters. This partitioning of the differential equations f ensures that away from events, $\underline{x}$ evolves according to $\dot{\underline{x}} = \underline{f}(x, y)$, whilst $\bar{x}$ and λ remain constant. Similarly, the partitioning of the reset equations h_j ensures that $\underline{x}$ and λ remain constant at reset events, but the discrete states $\bar{x}$ are reset to new values given by $\bar{x}^+ = \bar{h}_j(x^-, y^-)$.

Away from events, system dynamics evolve smoothly according to the familiar differential-algebraic model

$$\dot{x} = f(x, y) \tag{6}$$

$$0 = g(x, y) \tag{7}$$

where g is composed of $g^{(0)}$ together with appropriate choices of $g^{(i-)}$ or $g^{(i+)}$, depending on the signs of the corresponding elements of y_s. At switching events (3), some component equations of g change. To satisfy the new $g = 0$ constraints, algebraic variables y may undergo a step change. Impulse events (1) (alternatively reset events (4)) force a discrete change in elements of $\bar{x}$. Algebraic variables may again step to ensure $g = 0$ is always satisfied.

The *flows* of x and y are defined as

$$x(t) = \phi_x(x_0, t) \tag{8}$$

$$y(t) = \phi_y(x_0, t) \tag{9}$$

where $x(t)$ and $y(t)$ satisfy (1)-(3), along with initial conditions,

$$\phi_x(x_0, t_0) = x_0 \tag{10}$$

$$g\big(x_0, \phi_y(x_0, t_0)\big) = 0. \tag{11}$$

2.2 System theory

2.2.1 Marginally stable trajectories

Formulation of the proposed algorithm for computing marginally stable trajectories is based on some important assumptions:

A1. The limit set for the system is composed only of isolated points. (The limit set is the set of points to which the system converges in either forward or backward time.) This assumption excludes the existence of limit cycles, or other more exotic limit behaviour, for example Zeno effects.

A2. Trajectories ϕ_x, ϕ_y depend continuously on initial conditions (and hence parameters). This holds if:

- f, g of the DAE systems (6)-(7) that describe behaviour over periods between events are continuous with continuous first partial derivatives, and algebraic singularity is avoided [4],
- trajectories intersect event triggering hypersurfaces transversally [3, 5], and
- state mappings at events are continuous with continuous first partial derivatives.

These properties are true for many hybrid system application areas, including power systems. Furthermore, they ensure well defined trajectory sensitivities, which underlie the proposed algorithm. Trajectory sensitivities are considered briefly in Section 2.3.

Under those assumptions, the stability boundary is composed of the closure of the stable manifolds of type-1 unstable equilibrium points (UEPs) that lie on the boundary [6]. In other words, if a disturbance to the system is critically cleared, the trajectory will lie on the stable manifold of a type-1 UEP, and approach that UEP in infinite time. A trajectory that is nearly critically cleared, whether stable or unstable, will pass close to that UEP. But generally the UEP is unknown. Energy function methods require a knowledge of this 'controlling UEP' [1]. The proposed algorithm does not.

Equilibria of the system (6)-(7) satisfy $f(x, y) = 0$ with (7) necessarily satisfied everywhere. Therefore proximity to equilibria can be established using the quadratic cost function

$$\mathcal{J}(x, y) = \frac{1}{2} f(x, y)^t W f(x, y) \tag{12}$$

where W is a diagonal matrix used to weight the relative importance of the components of f. Near an equilibrium point, the cost $\mathcal{J}(x, y)$ becomes small. By monitoring $\mathcal{J}(x, y)$ along a trajectory, points that are local minima can be found. Many of these points indicate proximity to equilibria.

The proposed algorithm is based on the idea that $\mathcal{J}(x, y)$ defines a hypersurface in the variables t (time) and x_0 (initial conditions and parameters). Starting from a local minimum in $\mathcal{J}$ on the initial trajectory, t and x_0 can be varied to minimize $\mathcal{J}$ over that hypersurface. As $\mathcal{J}$ is minimized, the minimum point moves closer towards a UEP, and hence the trajectory moves closer to passing through that UEP. More complete details are given in Section 3. The examples of Section 4 illustrate these concepts.

It remains to determine the number of parameters that should be varied to move a trajectory so that it becomes coincident with the stability boundary. This is addressed in the following subsection.

2.2.2 Parameter dimension

Recall that $x \in \mathbb{R}^n$, so the state-space[1] has dimension n. The stable manifold of a type-1 UEP therefore has dimension $n - 1$. (It can be thought of as being defined by a single equation in n unknowns.) If p parameters are allowed to vary then the stable manifold, denoted $\mathcal{S}(x, \lambda)$, becomes an $(n + p - 1)$-manifold in $(n + p)$-space.

Now consider the system flow $\phi_x(x_0, t)$. For constant parameters, $\phi_x(x_0, t)$ is a 1-manifold, parameterized by time, that exists in state-space. The dimension of this manifold increases by one for each free parameter. Define $\mathcal{F}(x, \lambda)$ as the $(p+1)$-manifold, in $(n+p)$-space, that corresponds to p free parameters.

The intersection $\mathcal{S}(x, \lambda) \cap \mathcal{F}(x, \lambda)$ is therefore generically a p-manifold in $(n+p)$-space [7]. Now we desire this intersection to be a single isolated trajectory that lies on the stability boundary. So the intersection must be a 1-manifold, implying $p = 1$. Hence a single parameter should be varied in order to obtain a marginally stable system.

Comments:

- There is no guarantee that $\mathcal{S}$ and $\mathcal{F}$ will intersect for all choices of the single free parameter. However if they do intersect, then generically the intersection will be a single isolated trajectory that approaches the UEP.
- If more than one parameter is varied, then a continuum of marginally stable trajectories will usually result.

2.3 Trajectory sensitivities

Trajectory sensitivities provide a way of quantifying the variation of a trajectory due to small changes in parameters and/or initial conditions [8]. This section summarizes the main concepts. Complete details can be found in [3].

[1] The algebraic states y can be thought of as implicit functions of x, and so do not influence the state-space dimension.

To obtain the sensitivity of the flow ϕ_x, ϕ_y to initial conditions, and hence to parameter variations, the Taylor series expansion of (8)-(9) is formed. Neglecting higher order terms gives

$$\delta x(t) = \frac{\partial \phi_x(x_0, t)}{\partial x_0} \delta x_0 \triangleq \Phi_x(t) \delta x_0 \tag{13}$$

$$\delta y(t) = \frac{\partial \phi_y(x_0, t)}{\partial x_0} \delta x_0 \triangleq \Phi_y(t) \delta x_0. \tag{14}$$

It is important to keep in mind that x_0 incorporates λ, so sensitivity to x_0 includes sensitivity to λ. Equations (13)-(14) provide the changes $\delta x(t), \delta y(t)$ in a trajectory, at time t, for a given (small) change in initial conditions $\delta x_0 = [\delta \underline{x}_0^t \ \delta \bar{x}_0^t \ \delta \lambda^t]^t$.

Full details of the variational equations describing the evolution of the trajectory sensitivities for the DAIS model (1)-(3) are provided in [3]. An important conclusion is that trajectory sensitivities are defined for complex event driven behaviour. Furthermore, it is shown that the sensitivities can be calculated efficiently as a by-product of using an implicit numerical integration technique, such as trapezoidal integration, to produce the nominal system trajectory [3, 9].

3 Algorithm

3.1 Development

The algorithm is based on minimizing the quadratic cost function $\mathcal{J}(x, y)$ given by (12). Notice though from (8)-(9) that x and y are functions of x_0 and t. It was shown in Section 2.2.2 that this optimization problem has isolated solutions if a single parameter is varied. Therefore x_0 (the initial conditions and system parameters) is parameterized by a scalar parameter α. This parameterization is discussed further in Section 3.2.

The cost function $\mathcal{J}$ can be reformulated as

$$\begin{aligned} \mathcal{J}(\alpha, t) &= \frac{1}{2} f(\phi_x(x_0(\alpha), t), \phi_y(x_0(\alpha), t))^t W f(\phi_x(x_0(\alpha), t), \phi_y(x_0(\alpha), t)) \\ &= \frac{1}{2} \tilde{f}(\alpha, t)^t W \tilde{f}(\alpha, t). \end{aligned} \tag{15}$$

The aim of the algorithm is to find values of α and t that minimize (15).

Let $z = [\alpha \ t]^t$. Then (15) may be rewritten

$$\mathcal{J}(z) = \frac{1}{2} \tilde{f}(z)^t W \tilde{f}(z). \tag{16}$$

This unconstrained minimization can be achieved using a Gauss-Newton iterative method [10]. Iterations follow the scheme[2]

$$\left(\nabla \tilde{f}(z^k)^t W \nabla \tilde{f}(z^k) \right) \Delta z^k = \nabla \tilde{f}(z^k)^t W \tilde{f}(z^k) \tag{17}$$

$$z^{k+1} = z^k - a^k \Delta z^k \tag{18}$$

where a^k is an acceleration factor,

$$\nabla \tilde{f} = \begin{bmatrix} \dfrac{\partial \tilde{f}}{\partial \alpha} & \dfrac{\partial \tilde{f}}{\partial t} \end{bmatrix} \tag{19}$$

[2]Equation (17) could be solved directly by inverting $\nabla \tilde{f}^t W \nabla \tilde{f}$. However faster and more numerically robust algorithms are available [11].

and

$$\begin{aligned}\frac{\partial \tilde{f}}{\partial \alpha} &= \frac{\partial f}{\partial x}\frac{\partial \phi_x}{\partial x_0}\frac{\partial x_0}{\partial \alpha} + \frac{\partial f}{\partial y}\frac{\partial \phi_y}{\partial x_0}\frac{\partial x_0}{\partial \alpha} \\ &= (f_x\Phi_x + f_y\Phi_y)\frac{\partial x_0}{\partial \alpha} \qquad (20)\\ \frac{\partial \tilde{f}}{\partial t} &= \frac{\partial f}{\partial x}\frac{\partial \phi_x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial \phi_y}{\partial t} \\ &= f_x\dot{x} + f_y\dot{y}. \qquad (21)\end{aligned}$$

Notice the role played by trajectory sensitivities Φ_x, Φ_y in (20). The vector $\frac{\partial x_0}{\partial \alpha}$ is discussed in Section 3.2. By assumption, the algebraic equations (7) are satisfied everywhere, so that

$$\begin{aligned} & g_x\dot{x} + g_y\dot{y} = 0 \\ \Rightarrow\ & \dot{y} = -(g_y)^{-1}g_x\dot{x}.\end{aligned}$$

Substituting into (21) gives

$$\begin{aligned}\frac{\partial \tilde{f}}{\partial t} &= \left(f_x - f_y(g_y)^{-1}g_x\right)\dot{x} \\ &= \left(f_x - f_y(g_y)^{-1}g_x\right)\tilde{f}(\alpha, t). \qquad (22)\end{aligned}$$

To obtain $\tilde{f}(z^k)$ and $\nabla\tilde{f}(z^k)$, the simulation is run for a time t^k, where $z^k = [\alpha^k\ t^k]^t$. As mentioned in Section 2.3, the trajectory sensitivities required for $\nabla\tilde{f}$ are computed as a by-product of the simulation, and so introduce negligible extra computational cost. Equations (17)-(18) are then solved to update the estimate of α^k and t^k.

Iterations of the Gauss-Newton method cease when either $\tilde{f}(z^k) \approx 0$ or when $\nabla\tilde{f}(z^k)^t W\tilde{f}(z^k) \approx 0$. The former case corresponds to an equilibrium point lying on the trajectory. If that point is not the stable equilibrium point (SEP), then a marginally stable trajectory has been located, i.e., parameter values that cause the trajectory to run to a UEP have been identified. (Note that because a UEP can only be approached in infinite time, the optimal value of t given by the Gauss-Newton method will theoretically approach infinity. However in practice, iterations are halted before t grows too large.) It is possible that this UEP does not lie on the stability boundary. However it has been found that the algorithm usually converges to an appropriate type-1 UEP. This is a consequence of initiating the algorithm at a stable parameter set.

The second condition corresponds to local extrema. It is not clear that such points have any physical significance, though they certainly influence the convergence behaviour of the algorithm. It has been found that the algorithm can become trapped at such points, though often they are in close proximity to the desired UEP. (The examples of Section 4 illustrate this phenomenon.) Re-initialization at a slightly perturbed point is usually sufficient for progression to an improved solution.

3.2 Parameterization of x_0

The scalar parameter α describes the way in which x_0, the initial conditions and system parameters, are to be varied in order to find a marginally stable system trajectory. A single element of x_0 may be varied independently, in which case $\frac{\partial x_0}{\partial \alpha}$ is a vector of zeros except for a single nonzero entry in the appropriate location. Alternatively α may influence a few elements of x_0. Then $\frac{\partial x_0}{\partial \alpha}$ specifies a direction vector in x_0-space.

Typically, in studies of power system dynamic behaviour, the pre-disturbance system lies at a stable equilibrium point. Variation of α may well shift that equilibrium point. From (6)-(7), and using the fact that $f = [\underline{f}^t\ 0^t\ 0^t]^t$, the starting equilibrium point is described by

$$\begin{aligned} 0 &= \underline{f}(x_0(\alpha), y_0) \\ 0 &= g(x_0(\alpha), y_0).\end{aligned}$$

Differentiating gives

$$0 = \underline{f}_x \frac{\partial x_0}{\partial \alpha} d\alpha + \underline{f}_y dy_0$$
$$0 = g_x \frac{\partial x_0}{\partial \alpha} d\alpha + g_y dy_0$$

and further manipulation yields

$$0 = \left(\underline{f}_x - \underline{f}_y (g_y)^{-1} g_x \right) \frac{\partial x_0}{\partial \alpha} = \underline{f}^*_x \frac{\partial x_0}{\partial \alpha}. \tag{23}$$

Note that $\underline{f}^*_x$ is not square. Recall that $x = [\underline{x}^t \ \bar{x}^t \ \lambda^t]^t$, so (23) can be rewritten

$$0 = \underline{f}^*_{\underline{x}} \frac{\partial \underline{x}_0}{\partial \alpha} + \underline{f}^*_{\bar{x}} \frac{\partial \bar{x}_0}{\partial \alpha} + \underline{f}^*_{\lambda} \frac{\partial \lambda}{\partial \alpha}.$$

The $\underline{x}_0$ are dependent variables, whilst $\bar{x}_0$ and λ are independent variables. The required $\frac{\partial \underline{x}_0}{\partial \alpha}$ can be found using

$$\frac{\partial \underline{x}_0}{\partial \alpha} = -(\underline{f}^*_{\underline{x}})^{-1} \left(\underline{f}^*_{\bar{x}} \frac{\partial \bar{x}_0}{\partial \alpha} + \underline{f}^*_{\lambda} \frac{\partial \lambda}{\partial \alpha} \right) \tag{24}$$

where $\frac{\partial \bar{x}_0}{\partial \alpha}$ and $\frac{\partial \lambda}{\partial \alpha}$ are independent, and hence specified.

Note that for some choices of α, such as fault clearing time or machine inertia, $\frac{\partial \underline{x}_0}{\partial \alpha}$ will be zero. But for choices such as load level or generator mechanical torque, $\frac{\partial \underline{x}_0}{\partial \alpha}$ will reflect the change in the equilibrium point.

3.3 Variation of multiple parameters

As indicated in Section 2.2.2, variation of a single parameter results in an isolated marginally stable trajectory. Accordingly, variation of multiple parameters leads to a continuum of marginally stable trajectories. The proposed algorithm cannot directly compute such a continuum. However it can be embedded within higher level applications that deal with multiple parameter variation. These include:

Continuation methods. If two parameters are allowed to vary, the set of parameters that induce marginal stability forms a curve (1-manifold) in parameter space. Such curves are used widely in power system operations, where they have become known as nomograms. The algorithm proposed in Section 3.1 finds points along that curve. Simple parameter variation strategies can be used to move from point to point around such a curve.

Optimization. Variation of more than two parameters is typically associated with dynamic-embedded optimization problems, where the objective is to maximize or minimize a cost function subject to marginal stability of the system. In the context of power systems, this type of problem arises, for example, when determining minimum cost generation schedules of stability-constrained systems. Solution of these problems involves coupling the marginal stability algorithm with a strategy for determining a cost-minimizing direction.

4 Examples

The algorithm of Section 3 is illustrated using a power system of the form shown in Figure 1. Both generators were represented by a two axis (fourth order) machine model [12]. In all cases, the system was disturbed by the application of a three phase fault at the terminal bus of Generator 1.

Two studies are presented. In the first, the calculation of the critical fault clearing time is considered. The second study explores the maximum loading of the generators.

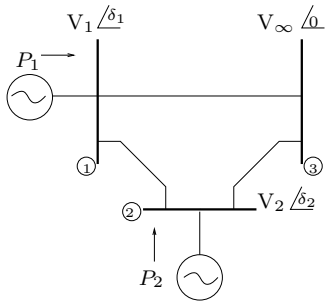

Figure 1: Two machine infinite bus system.

Initial guess (sec)	Iterations (sec)			Actual value (sec)
	1	2	3	
0.4000	0.3011	0.2940	0.2931	0.2866

Table 1: Estimates of t_f.

4.1 Critical clearing time

For this study, the fault-on time t_f was chosen as the unknown parameter α that was varied to minimize the cost function $\mathcal{J}$ given by (15). The fault switching mechanism, and hence t_f, was incorporated into the DAIS model (1)-(3), by introducing the equations

$$\dot{x}_t = 1, \qquad x_{t,0} = 0$$
$$0 = (x_t - t_{init})(x_t - (t_{init} + t_f)) - y_s$$

where x_t is a timer, and t_{init} is the time at which the fault is initiated. The algebraic variable y_s is negative during the fault-on period, and positive otherwise. It is used to switch an algebraic equation, as in (3). When y_s is negative, fault current is added to the net current injection at the faulted bus. Using this formulation, t_f can be treated like any other parameter. Variation of t_f does not affect other initial conditions though, so $\frac{\partial x_0}{\partial \alpha}$ in (20), (23) is a vector of zeros except for a 1 in the location corresponding to t_f.

In this investigation an initial guess of $t_f = 0.40$ sec was used. It can be seen from the phase portrait of Figure 2 that the system was clearly unstable for this initial guess. Subsequent Gauss-Newton iterations using (17)-(18) are given in Table 1. The final estimate of $t_f = 0.2931$ sec compares well with the actual critical value of $t_f^* = 0.2866$ sec obtained by repetitive simulation. The iterations are illustrated in Figure 2. (Note that this figure is only a two dimensional projection of the eight dimensional state-space.) The figure shows the trajectories corresponding to the initial guess, the iterations and final estimate, and the actual critical value.

Figure 2 shows that behaviour becomes rather complicated as the critical parameter value is approached. This is reflected in the cost function $\mathcal{J}$ as multiple local minima. In this case the algorithm converged to a nearby local minimum rather than the desired minimum corresponding to the UEP. However from a practical perspective, the error in the final estimate of t_f is inconsequential, as accuracy to two decimal places is more than sufficient. If desired, an improved final estimate could be obtained by restarting the iterative solution process.

Further illustration is provided by Figure 3, where the cost function $\mathcal{J}$ is plotted against time. It is clear that for the initial parameter guess, $\mathcal{J}$ had only a single minimum, near 1 sec. However as the parameter estimate was progressively improved, a local minimum developed near 1.5 sec. The solution process in this case locked onto that local minimum. By restarting the process from the deeper minimum at 2.4 sec, an improved estimate of $t_f = 0.2878$ sec could be obtained.

A simple process for choosing the acceleration factor a was used throughout this sequence of studies. A more sophisticated choice of a may well have resulted in direct convergence to the improved estimate of t_f. Further investigations are required.

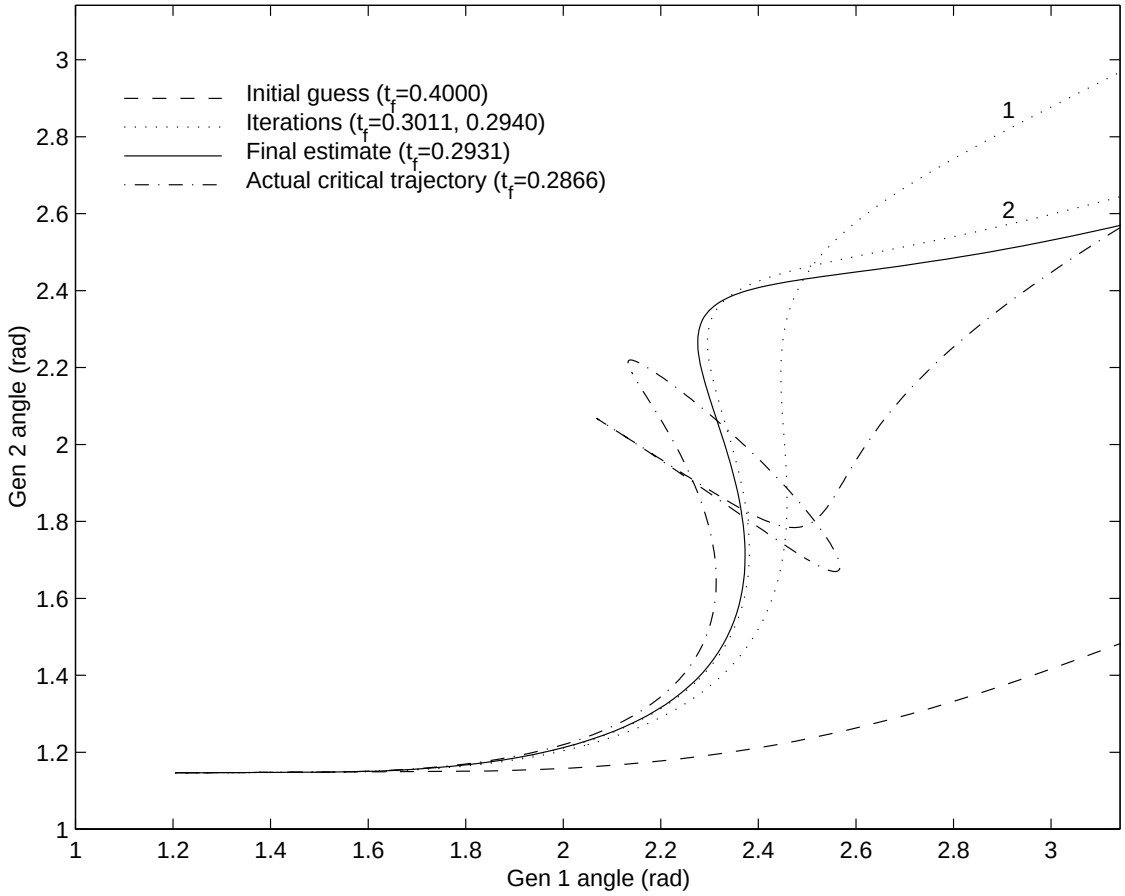

Figure 2: Phase portrait view of trajectories, study A.

4.2 Critical generator loading

The aim of this study was to determine the values of generator mechanical torques T_m that corresponded to the system being marginally stable, given the fault scenario described earlier. In this case the fault-on time was held fixed at 0.3 sec. This study therefore effectively determined the maximum power that could be delivered by the generators whilst still ensuring stability for the specified fault condition.

An initial guess of $T_m = 0.72$ pu was used for both generators. Also, T_m was varied in unison at the two generators during the iterations. (This was not a limitation of the algorithm, but rather assisted in presentation clarity.) The phase portrait of Figure 4 shows the system response for the base case. The system was clearly stable. The Gauss-Newton algorithm proceeded to estimate a critical value of $T_m = 0.7934$ pu. Repeated simulations indicated that the actual critical value (to four decimal places) was $T_m = 0.7930$ pu. Figure 4 shows the trajectories corresponding to the estimated and actual critical values. There is clear propinquity.

Notice in Figure 4 that the trajectories for $T_m = 0.72$ pu and $T_m = 0.7934$ pu start at different points. (The starting points are marked by a '$\circ$'.) This occurs because variation of T_m alters the operating point, i.e., the initial point $\underline{x}_0$. The dependence of $\underline{x}_0$ on the parameter α is reflected through $\frac{\partial \underline{x}_0}{\partial \alpha}$, in accordance with (24).

Convergence from the initial guess for T_m to the final estimate took 11 iterations, though convergence of T_m was virtually achieved in 3 iterations. The iterations are shown by '$\times$' in the contour plot of Figure 5. This plot shows contours of the cost function $\mathcal{J}$ for various values of T_m and time t. (Recall from (15) that $\mathcal{J}$ is a function of α and t, with α in this case driving

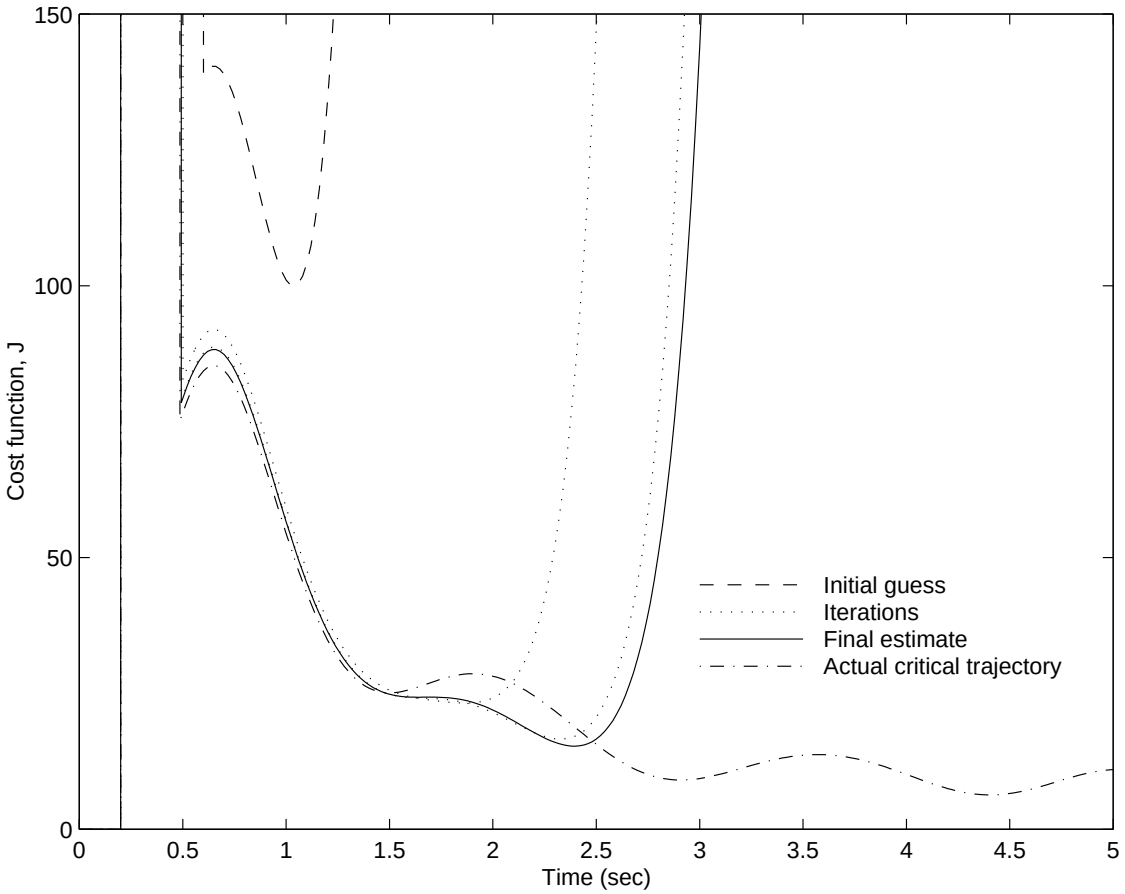

Figure 3: Cost function $\mathcal{J}$ versus time.

T_m.) An interesting decoupling can be observed in the convergence process. Over the first three iterations, T_m made significant progress towards its final value, reaching 0.7980 pu compared with the critical value of 0.7930 pu. On the other hand, the optimization variable t changed little over the initial iterations, but experienced greater change over the latter phase. As T_m is the variable of primary interest, with t only playing an auxiliary role, convergence could be truncated after the third iteration.

It is interesting to note that the first iteration step is very small. This is because the cost function is quite flat near the initial parameter guess. In fact that initial point in near a saddle in $\mathcal{J}$. If the algorithm was started from a guess of $T_m \approx 0.7$ pu, subsequent iterations would move away from the desired minimum, towards a minimum corresponding to the stable equilibrium point.

5 Conclusions

The paper develops an iterative approach to finding parameter values that result in marginally stable system behaviour for a specified disturbance. This approach is based on the fact that trajectories which lie on the stability boundary must approach an unstable equilibrium point. The algorithm varies parameters to minimize a cost function which is a measure of the distance from equilibria. By minimizing this cost function, trajectories move closer to passing through a UEP, and hence move closer to lying on the stability boundary.

Two studies have illustrated the algorithm. In the first, the fault-on time was chosen as the

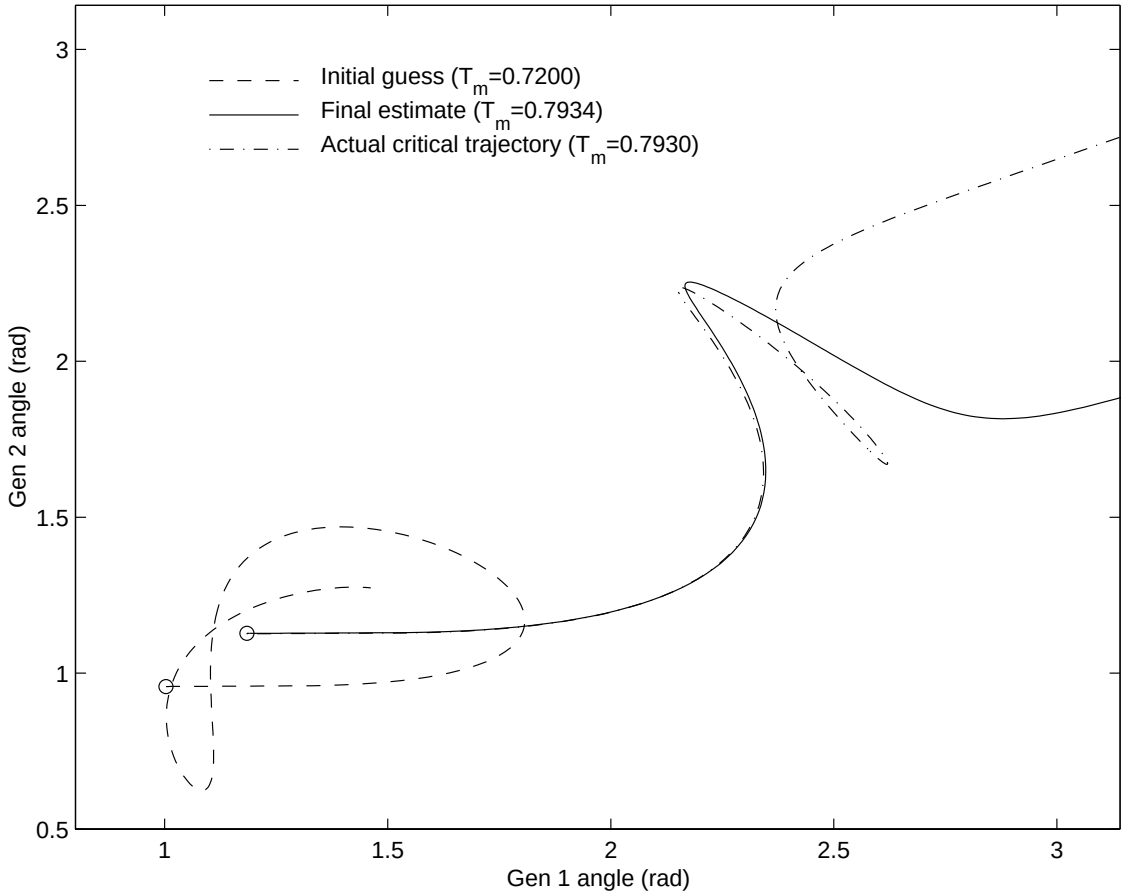

Figure 4: Phase portrait view of trajectories, study B.

parameter of interest. The second study found the generator loading condition that corresponded to the transient stability limit. Rapid computation of such dynamically-induced maximum loading conditions allows more accurate assessment of the risks of instability. On-line implementation would assist in safely maximizing power system utilization, with consequent economic benefits.

Applications that involve variation of multiple parameters can be built from the marginal stability algorithm. Extensions required for continuation methods and optimization problems have been considered in the paper.

References

[1] M.A. Pai, *Energy Function Analysis for Power System Stability*, Kluwer Academic Publishers, Boston, MA, 1989.

[2] I.A. Hiskens, "Power system modeling for inverse problems," *IEEE Transactions on Circuits and Systems I*, To appear.

[3] I.A. Hiskens and M.A. Pai, "Trajectory sensitivity analysis of hybrid systems," *IEEE Transactions on Circuits and Systems I*, vol. 47, no. 2, pp. 204–220, February 2000.

[4] D.J. Hill and I.M.Y. Mareels, "Stability theory for differential/algebraic systems with application to power systems," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 11, pp. 1416–1423, November 1990.

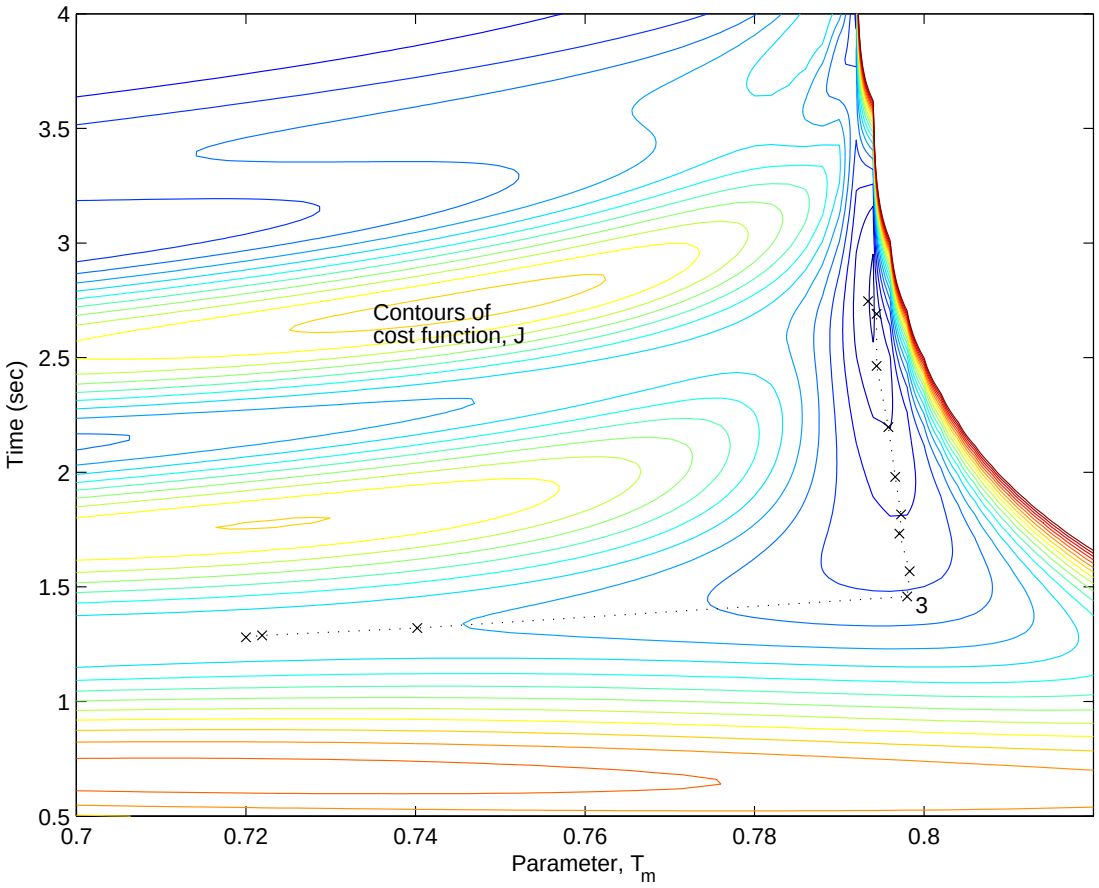

Figure 5: Contours of cost function $\mathcal{J}$ versus T_m and t.

[5] M.S. Branicky, V.S. Borkar, and S.K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, January 1998.

[6] H-D. Chiang, M.W. Hirsch, and F.F. Wu, "Stability regions of nonlinear autonomous dynamical systems," *IEEE Transactions on Automatic Control*, vol. 33, no. 1, pp. 16–27, January 1988.

[7] V.W. Guillemin and A. Pollack, *Differential Topology*, Prentice Hall, Englewood Cliffs, NJ, 1974.

[8] P.M. Frank, *Introduction to System Sensitivity Theory*, Academic Press, New York, 1978.

[9] W.F. Feehery, J.E. Tolsma, and P.I. Barton, "Efficient sensitivity analysis of large-scale differential-algebraic systems," *Applied Numerical Mathematics*, vol. 25, pp. 41–54, 1997.

[10] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.

[11] G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins, 2nd edition, 1989.

[12] P.W. Sauer and M.A. Pai, *Power System Dynamics and Stability*, Prentice Hall, Upper Saddle River, NJ, 1998.