Bulk Power System Dynamics and Control V, August 26-31, 2001, Onomichi, Japan

Inverse Problems in Power Systems

I.A. Hiskens[1]
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana IL 61801 USA

Abstract— **Large disturbances in power systems often initiate complex interactions between continuous dynamics and discrete events. Such behaviour can be modeled in a systematic way by a set of differential-algebraic equations, modified to incorporate impulse (state reset) action and constraint switching. The paper presents a practical object-oriented approach to implementing the DAIS model. The systematic nature of the DAIS model enables efficient computation of trajectory sensitivities, which in turn facilitate algorithms for solving inverse problems. The paper outlines a number of inverse problems, including parameter uncertainty, parameter estimation, boundary value problems, border-collision bifurcations, locating critically stable trajectories, and optimal control.**

Keywords: Dynamic modeling, power system dynamics, hybrid systems, inverse problems.

I. Introduction

Interactions between continuous dynamics and discrete events are an intrinsic part of power system dynamic behaviour. Devices that obey physical laws typically exhibit continuous dynamics. Examples range from generators and their controllers at the system level, through to capacitors and inductors within power electronic circuits. On the other hand, event-driven discrete behaviour is normally associated with rule-based components. Examples in this latter category include protection devices [1], tap-changing transformers [2], power electronic switches [3] and supervisory control [4]. Limits within physical devices also fall into this category; an event occurs when a controller signal saturates or a FACTS device encounters its maximum/minimum firing angle.

To illustrate continuous/discrete interactions in power systems, consider a disturbance consisting of an initiating event, such as a lightning strike on a transmission line, followed by protection action to remove the fault. The fault would disturb the steady-state balance between the electrical and mechanical torques on generator shafts, causing angles and frequencies to respond dynamically. In parallel, protection relays should detect the fault and decide on the appropriate response. Trip signals are sent to circuit breakers, which disconnect the faulted feeder after a small (mechanical) time delay. Meanwhile, oscillations induced in intermachine angles may or may not be stable. Removal of the faulted line could lead to overloading of other feeders, and their subsequent tripping. The consequent increased demand for reactive power may activate generator over-excitation protection, causing a reduction in terminal voltage, increased system losses, further overloading of feeders and finally system disintegration. Whilst this scenario seems pessimistic, it has occurred, to the detriment (and annoyance) of many consumers!

Similar continuous/discrete interactions exist across all layers of power systems. At the market layer, for example, system measurements and participant inputs are interpreted in terms of market rules to generate events that affect the physical system.

In all cases, discrete events influence continuous dynamics, which in turn trigger new events. Modeling and simulation must accurately capture these interactions. Power system simulation has generally evolved to the point where the continuous/discrete nature of dynamic behaviour is fairly accurately replicated. However it is common to find event handling treated as an *ad hoc* addition to continuous state simulation. The nature of inverse problems dictates a more systematic *hybrid systems* approach to capturing continuous/discrete interactions.

Power system analysis normally addresses *forward problems*. Given a system model and a set of parameters, system response can be determined. However the disturbance scenario outlined above motivates analysis questions that are classed as *inverse problems* [5]. Such a disturbance would generate recordings from Wide Area Measurement Systems [6]. Those measurements could be used to improve estimates of parameters of system models [7], [8]. This is an inverse problem; the measured response is given, and a model is used to infer parameter values. It's easy to postulate other inverse problems. For example, what are the minimal changes in controllable parameters, perhaps generator MW and/or voltage setpoints, that would avoid cascading tripping of overloaded feeders or voltage dip problems or instability? How significantly do certain parameters, e.g., load voltage dependence, impact system behaviour?

Traditionally such inverse problems have been addressed by repeated solution (simulation) of forward problems. However systematic modeling provides the foundation for algorithms that address inverse problems directly.

The paper has the following structure. Section II presents a systematic representation of hybrid systems that is applicable for analysis of power system dynamics. Implementation issues are discussed in Section III. Inverse problems are considered in Section IV, and conclusions are presented in Section V.

[1]Research supported by the EPRI/DoD Complex Interactive Networks/Systems Initiative, the National Science Foundation through the grant ECS-0085755, and the Grainger Foundation.

II. Hybrid System Representation

A. Background

Power systems are an important example of hybrid systems, which are characterized by:

- continuous and discrete states;
- continuous dynamics;
- discrete events, or triggers;
- mappings that define the evolution of discrete states at events.

Conceptually such systems can be thought of as an indexed collection of continuous dynamical systems

$$F_q(\dot{x}, x, y) = 0, \tag{1}$$

along with a mechanism for 'jumping' between those systems, i.e., for switching between the various F_q. Each system is indexed by the discrete state q, whilst x and y are the continuous dynamic and algebraic states respectively. The jumping reflects the influence of the discrete event behaviour, and is dependent upon both a trigger condition and a discrete state evolution mapping. Overall system behaviour can be viewed as a sequential patching together of dynamical systems, with the final state of one dynamical system specifying the initial state for the next.

B. Simulation model

Various abstract models, such as Petri nets and hybrid automata [9], [10], [11], provide a framework for establishing rigorous mathematical representations of physical devices and systems. However those representations are not immediately applicable to forward problems (via simulation), much less inverse problems. A model that captures the full richness of hybrid system behaviour, yet has a form suitable for simulation, is required.

Simulation techniques and properties are well established for differential-algebraic (DAE) systems [12]. Therefore the proposed hybrid system model is adapted from that basic form by incorporating impulsive action and switching of algebraic equations, giving the *DA Impulsive Switched* (DAIS) model

$$\dot{x} = f(x,y) + \sum_{j=1}^{e} \delta(y_{e,j})\Big(h_j(x,y) - x\Big) \tag{2}$$

$$0 = g(x,y) \equiv g^{(0)}(x,y) + \sum_{i=1}^{d}\Big(g^{(i-)}(x,y) + u(y_{d,i})\Big(g^{(i+)}(x,y) - g^{(i-)}(x,y)\Big)\Big) \tag{3}$$

where

- $x \in \mathbb{R}^n$ are dynamic states, and $y \in \mathbb{R}^m$ are algebraic states;
- $\delta(.)$ is the Dirac delta;
- $u(.)$ is the unit-step function;
- $f, h_j : \mathbb{R}^{n+m} \to \mathbb{R}^n$;
- $g^{(0)}, g^{(i\pm)} : \mathbb{R}^{n+m} \to \mathbb{R}^m$; some elements of each $g^{(.)}$ will usually be identically zero, but no elements of the composite g should be identically zero; the $g^{(i\pm)}$ have the same general form as g and are defined via (3), leading to a recursive structure for g;
- y_d, y_e are selected elements of y that trigger algebraic switching and state reset (impulsive) events respectively; y_d and y_e may share common elements.

The impulse and unit-step terms of the DAIS model can be expressed in alternative forms:

- Each impulse term of the summation in (2) can be expressed in the state reset form

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

  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 $h'_j(x^+, x^-, y^-) = 0$.
- The contribution of each $g^{(i\pm)}$ in (3) can be expressed as

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

  with (3) becoming

$$0 = g(x,y) \equiv g^{(0)}(x,y) + \sum_{i=1}^{d} g^{(i)}(x,y). \tag{5}$$

  This form is often more intuitive than (3).

Equations (2),(3) are a reformulation (and slight generalization) of the model proposed in [13].

It can be convenient to establish the partitions

$$x = \begin{bmatrix} \underline{x} \\ z \\ \lambda \end{bmatrix}, \qquad f = \begin{bmatrix} \underline{f} \\ 0 \\ 0 \end{bmatrix}, \qquad h_j = \begin{bmatrix} \underline{x} \\ \underline{h}_j \\ \lambda \end{bmatrix} \tag{6}$$

where

- $\underline{x}$ are the continuous dynamic states, for example generator angles, velocities and fluxes;
- z are discrete dynamic states, such as transformer tap positions and protection relay logic states;
- λ are parameters such as generator reactances, controller gains and switching times.

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 z 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 dynamic states z are reset to new values given by $z^+ = \underline{h}_j(x^-, y^-)$.

Remark:

- The DAIS model assumes constant state space dimension $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ across events. This differs from some other hybrid system implementations, for example [14], where the state dimension is allowed to vary upon component switching. The DAIS formulation is not restrictive, though it may require carrying some

'inactive' states following an event. Maintaining constant state dimension has a number of advantages: 1) the variational equations describing trajectory sensitivities, presented in Appendix A, have a simpler form, and 2) switched states are more easily incorporated into objective functions of optimization-based inverse problems.

Initial conditions for the model (2)-(3) are given by $x(t_0) = x_0$ and $y(t_0) = y_0$, where y_0 is a solution of $g(x_0, y_0) = 0$. Note that in solving for y_0, the constraint switching described by (3) must be taken into account. This establishes the initial discrete state q_0.

The *flows* of x and y are defined as

$$\phi(x_0, t) = \begin{bmatrix} \phi_x(x_0, t) \\ \phi_y(x_0, t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \tag{7}$$

where $x(t)$ and $y(t)$ satisfy (2)-(3), along with initial conditions,

$$\phi_x(x_0, t_0) = x_0 \tag{8}$$

$$g(x_0, \phi_y(x_0, t_0)) = 0. \tag{9}$$

C. Trajectory sensitivities

Trajectory sensitivities provide a way of quantifying the variation of a trajectory resulting from (small) changes to parameters and/or initial conditions [13], [15]. To obtain the sensitivity of the flows ϕ_x and ϕ_y to initial conditions x_0, the Taylor series expansion of (7) is formed. Neglecting higher order terms gives

$$\Delta x(t) = \frac{\partial x(t)}{\partial x_0} \Delta x_0 \equiv x_{x_0}(t) \Delta x_0 \tag{10}$$

$$\Delta y(t) = \frac{\partial y(t)}{\partial x_0} \Delta x_0 \equiv y_{x_0}(t) \Delta x_0. \tag{11}$$

Using the partitioning (6), x_0 incorporates parameters λ, so sensitivity to initial conditions x_0 includes parameter sensitivity. Equations (10)-(11) describe the changes $\Delta x(t)$ and $\Delta y(t)$ in a trajectory, at time t along the trajectory, for a given (small) change in initial conditions $\Delta x_0 = [\Delta \underline{x}_0^t \quad \Delta z_0^t \quad \Delta \lambda^t]^t$. The time-varying partial derivatives x_{x_0} and y_{x_0} are known as *trajectory sensitivities*. An overview of the variational equations describing the evolution of these sensitivities is provided in Appendix A.

Along smooth sections of the trajectory, the trajectory sensitivities evolve according to a linear time-varying differential-algebraic system (20)-(21). For large systems, these equations have high dimension. However the computational burden is minimal when an implicit numerical integration technique such as trapezoidal integration is used to generate the trajectory. An overview of this result is provided in Appendix B. More complete details of both appendices are given in [13].

III. Implementation

A. Flexible component interconnection

Models of large systems are most effectively constructed using a hierarchical or modular approach. With such an approach, components are grouped together as subsystems, and the subsystems are combined to form the full system. This allows component and subsystem models to be developed and tested independently. It also allows flexibility in interchanging models.

The interactions inherent in hybrid systems are counter to this decomposition into subsystems and components. However the algebraic equations of the DAIS model can be exploited to achieve the desired modularity. Each component or subsystem can be modeled autonomously in the DAIS structure, with 'interface' quantities, e.g., inputs and outputs, established as algebraic variables. The components are then interconnected by introducing simple algebraic equations that 'link' the interface variables. This is similar to the connections concept of [16]. Note that all interconnections are non-causal [17], i.e., no rigid input-output arrangement of components is assumed.

To illustrate this linking concept, consider a case where the n-th algebraic state of component j, denoted $y_{j,n}$, is required by component k. In the model of component k, the corresponding quantity would appear as an algebraic variable $y_{k,m}$. The connection is made via the simple algebraic equation $y_{j,n} - y_{k,m} = 0$. In general, all linking can be achieved by summations of the form

$$\sum c_k y_{i,j} = 0 \tag{12}$$

where c_k is ± 1. Notice that all connections are external to the component models.

The linking strategy results in an interesting structure for the Jacobian[2]

$$J = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix}.$$

Components contribute square blocks down the diagonal of f_x and flattened rectangular blocks along the diagonal of the upper section of g_y. The lower section of g_y is an incidence matrix, with ± 1's given by the external connections (12). Figure 1 illustrates this structure. (This particular matrix corresponds to a ten generator representation of the Nordel power system [8].) A Jacobian structure like that of J was identified in [18], where a similar arrangement of components and connections was used in the development of an optimal power flow.

The structure and values of the lower connection submatrix of J, and hence $F_{\mathscr{x}}$, are fixed for all time. This can be exploited in the factorization of $F_{\mathscr{x}}$ to improve the efficiency of solving (26)-(27). The efficiency improvement can be significant, as these equations are solved at every time step.

In general, components and subsystems of any form can be modeled, provided they are structured with interfacing algebraic variables that can be linked to other components. Noise and/or random disturbances can be added to the model by linking components that generate random signals.

[2]This Jacobian has the same structure as the matrix $F_{\mathscr{x}}$, given by (28), that is required for implicit numerical integration and for computing trajectory sensitivities.

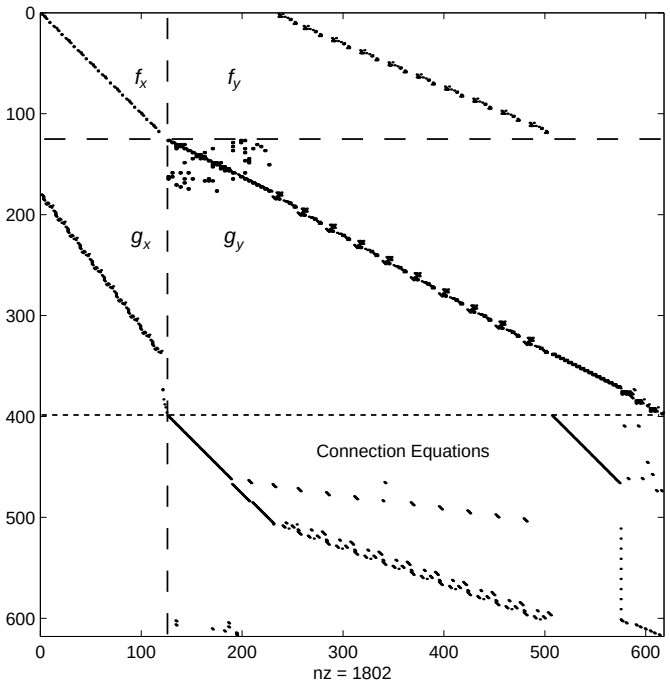

Fig. 1. Sparsity structure of Jacobian J.

B. Matlab implementation

The proposed modular approach to constructing hybrid systems has been implemented in Matlab [19]. In this implementation, the system is described by a data file that contains `model_data` information and (separate) `connections` details. Each component of the system contributes an entry to `model_data` that consists of the component name, initial values for x_0 and y_0, and background parameters. Links between components are fully described in `connections` using the form given by (12).

Every component is described by a file that calculates values for f, g and h, and sparsely stored elements of the partial derivative matrices f_x, f_y, g_x, g_y, h_x and h_y. These component files are re-usable, i.e., case independent, and reside in a component library. Relative indexing is used for Jacobian elements, as each component model is autonomous. (All connection information is externally defined.) Hence within a model, the indexing of Jacobian elements uses only local equation and variable numbering. The simulation kernel uses these relative indices, along with knowledge of equation and variable dimensions across all models, to generate the location of each element in the full matrices, i.e., the absolute indices. The actual matrices are never built explicitly, but rather are stored sparsely.

C. Symbolic differentiation

As indicated above, partial derivative matrices are calculated and stored sparsely by component files. Hand derivation of these partial derivatives can be tedious for large complicated models. Therefore the process has been automated through the use of symbolic differentiation [19]. Symbolic manipulation has been utilized in power system simulation previously [20], though the implementation was quite different.

The generation of a component file begins with an analytical model in the DAIS form. The analytical model must be unambiguously mapped into a character representation that can be manipulated symbolically. It is also important that this mapping does not restrict the implementation of the DAIS form. Fortunately the DAIS model structure is well suited to such translation. All elements of the model can be clearly and uniquely identified.

A Matlab function has been developed for translating the input model representation into a component file that can interact with the simulation kernel [19]. Building the f, g and h equations involves relatively straightforward character string manipulation. Generating the partial derivatives is more challenging. Firstly, equations and variable strings are converted to symbols. Symbolic differentiation produces partial derivatives that must be simplified and converted back to strings. If the final expression is zero, the derivative is discarded, as matrices are stored sparsely.

Component files are generated off-line and stored in the component library. Therefore symbolic manipulation does not slow simulation speed.

D. Computation of junction points

Switching and reset events generically do not coincide with the time instants of the numerical integration process. However simulation accuracy depends upon correct location, between integration time steps, of events [21].

A simple check of sign changes in trigger variables y_d and y_e at each integration step will reveal most events [13]. However this check fails to detect events where the associated trigger variables change sign an even number of times over a time step. A more thorough search for events is required, though a trade-off must be made between search accuracy and computational cost. An efficient approach proposed in [21] uses interpolation polynomials generated by a BDF integration method [12].

IV. Inverse Problems

System analysis is often tantamount to understanding the influence of parameters on system behaviour, and applying that knowledge to achieve a desired outcome. The 'known' information is the desired outcome. The parameters that achieve that outcome must be deduced. Because of the inverse nature of the problem, the process has traditionally involved repeated simulation of the model. This can be time consuming and frustrating, as the relationship between parameters and behaviour is often not intuitively obvious.

Systematic modeling, as presented in previous sections, allows the development of new tools that can solve inverse problems directly, albeit via iterative techniques. The DAIS model is conducive to the efficient generation of trajectory sensitivities. Those sensitivities quantify, to first order, the effects of parameters on dynamic behaviour. They

therefore underlie the development of gradient-based algorithms.

The following subsections present a range of inverse problems. Algorithms that address those problems are outlined. This list is not exhaustive, but seeks to provide an overview of the possibilities that follow from systematic modeling.

A. Parameter uncertainty

System parameters can never be known exactly. In fact uncertainty in some parameters, e.g., load models, can be quite high. Quantifying the effects of parameter uncertainty is not strictly an inverse problem, but illustrates the usefulness of trajectory sensitivities.

Because of the uncertainty in parameters, investigation of system behaviour should (ideally) include multiple studies over a range of parameter values. However simulation of large systems is computationally intensive. Such an investigation would be extremely time consuming. The more practical approach is to assume that a *nominal* set of parameters provides an adequate representation of behaviour over the full range of values. This may not always be a good assumption though.

A computationally feasible (though approximate) approach to repeated simulation is to generate a first order approximation of the trajectory for each set of perturbed parameters. The first order approximation is obtained by truncating the Taylor series expansion of the flow ϕ. Using (10)-(11), this gives

$$\phi(x_{0_2}, t) = \phi(x_{0_1}, t) + \begin{bmatrix} x_{x_0}(t) \\ y_{x_0}(t) \end{bmatrix} (x_{0_2} - x_{0_1}) \qquad (13)$$

where $x_{x_0}(t)$, $y_{x_0}(t)$ are computed along the nominal trajectory $\phi(x_{0_1}, t)$. Therefore if the trajectory sensitivities $x_{x_0}(t)$, $y_{x_0}(t)$ are available for a nominal trajectory, then (13) can be used to provide a good estimate of trajectories $\phi(x_{0_2}, t)$ corresponding to other (nearby) parameter sets. (Recall that parameters λ are embedded in x_0.)

The computational burden involved in generating the approximate trajectories is negligible. Given the nominal trajectory and associated trajectory sensitivities, new (approximate) trajectories can be obtained for many parameter sets. Therefore a Monte-Carlo technique can be employed to quantify the uncertainty in a trajectory:

- parameter sets are randomly generated,
- first order approximations are obtained using (13).

Figure 2 illustrates this process for a simple example where a disturbance initiates interactions between a tap-changing transformer and a dynamic load. The dark line shows the nominal trajectory. The bound around that trajectory was obtained using 200 randomly chosen sets of parameters. Further details can be found in [22].

Statistics quantifying the uncertainty in system behaviour due to parameter uncertainty can be obtained from the Monte-Carlo simulation. For example, it's possible to state the probability that a disturbance would initiate protection operation or that a voltage would fall below some predetermined threshold.

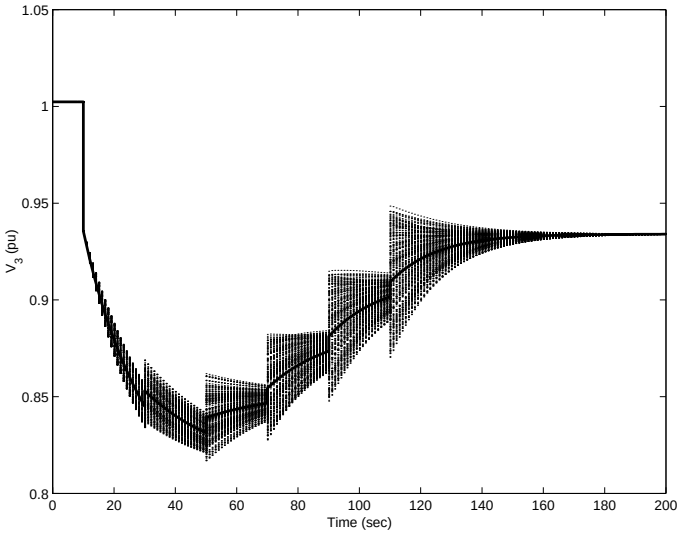

Fig. 2. Trajectory bounds.

Another approach to assessing the significance of parameter uncertainty is via worst case analysis [23]. This involves finding the values of parameters (within specified bounds) that induce the greatest deviation in particular system variables, for example voltages. The algorithm can be formulated as a constrained optimization, and is truly an inverse problem. Such optimization problems are discussed as part of later inverse problems.

B. Parameter estimation

System-wide measurements of power system disturbances are frequently used in event reconstruction to gain a better understanding of system behaviour [7], [24]. In undertaking such studies, measurements are compared with the behaviour predicted by a model. Differences are used to tune the model, i.e., adjust parameters, to obtain the best match between the model and the measurements. This process requires a systematic approach to,

1) identifying *well-conditioned* parameters that can be estimated reliably from the available measurements, and
2) obtaining a best estimate for those parameters.

It is shown in [8] that trajectory sensitivities can be used to guide the search for well-conditioned parameters, i.e., parameters that are good candidates for reliable estimation. Large trajectory sensitivities imply the corresponding parameters have leverage in altering the model trajectory to better match the measured response. Small trajectory sensitivities, on the other hand, imply that large changes in parameter values would be required to significantly alter the trajectory. Parameters in the former category are well-conditioned, whereas the latter parameters are ill-conditioned. Only parameters that influence measured states can be identified. A parameter may have a significant influence on system behaviour, but if that influence is not observable in the measured states, then the

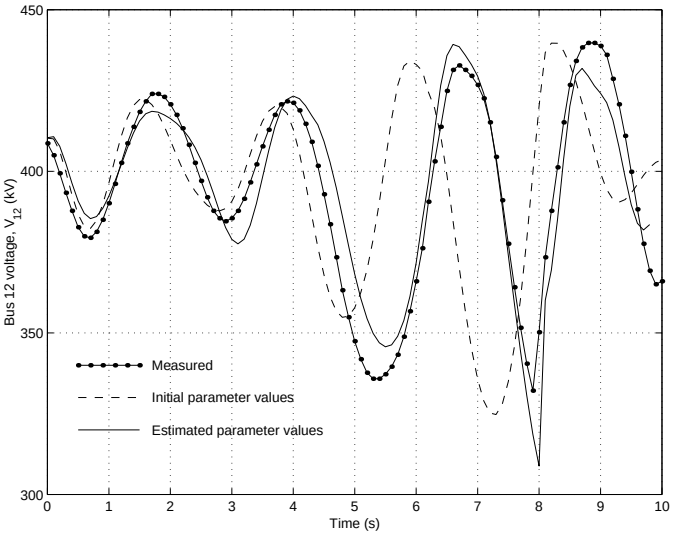

Fig. 3. Parameter estimation.

parameter is not identifiable. The concept of identifiability is explained more formally in [25].

A parameter estimation algorithm that is based on a Gauss-Newton iterative procedure is presented in [8]. The algorithm minimizes a nonlinear least-squares cost

$$\mathcal{V}(\theta) = \frac{1}{2}\|\breve{y}(\theta) - ms\|_2^2$$

where ms are the sampled measurements of the disturbance, $\breve{y}(\theta)$ are the flows provided by the model that correspond to the measured quantities, and θ are the unknown parameters. This minimization can be achieved (locally at least) by the iterative scheme

$$\begin{aligned} S(\theta^j)^t S(\theta^j)\Delta\theta^{j+1} &= S(\theta^j)^t(\breve{y}(\theta^j) - ms) \\ \theta^{j+1} &= \theta^j - \alpha^{j+1}\Delta\theta^{j+1} \end{aligned} \qquad (14)$$

where α^{j+1} is a scalar that determines the parameter update step size[3]. The matrix S is built from the trajectory sensitivities $\breve{y}_\theta$, i.e., sensitivity of model flows $\breve{y}$ to parameters θ. The invertibility of S^tS relates directly to identifiability [25].

The parameter estimation process is illustrated in Figure 3. A voltage measurement from a disturbance on the Nordel system [8] is shown. The figure also shows the simulated voltage trajectory for the initial parameter values, and the tuned values obtained after convergence (in four iterations) of (14). The improvement is clear.

Remarks:

1) Parameter estimation via (14) is not restricted to smooth systems. In fact, it is possible to estimate parameters that underlie event descriptions (provided measurements capture an occurrence of the event.)
2) For large systems, feasibility of the Gauss-Newton algorithm is dependent upon efficient computation of trajectory sensitivities. This underlines the importance of systematic modeling.

[3]Equation (14) could be solved by inverting S^tS, however faster and more numerically robust algorithms are available [26].

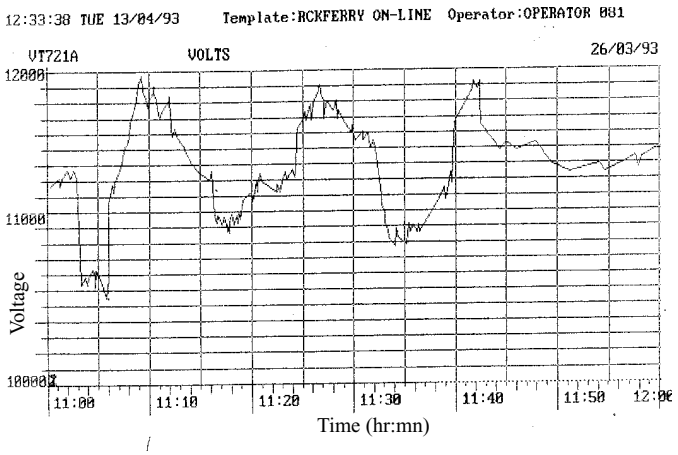

Fig. 4. Oscillations in distribution system voltage.

C. Boundary value problems

It is interesting to consider boundary value problems of the form

$$r\big(x_0, x(t_f)\big) = 0 \qquad (15)$$

where t_f is the final time, and $x(t)$ is the trajectory that starts from x_0 and is generated by (2),(3). The initial values x_0 are variables that must be adjusted to satisfy r. (Though r may directly constrain some elements of x_0.) To establish the solution process, (15) can be rewritten

$$r\big(x_0, \phi_x(x_0, t_f)\big) = 0, \qquad (16)$$

which has the form $\tilde{r}(x_0) = 0$. Boundary value problems can be solved by shooting methods [27], [28], which are a combination of Newton's method for solving (16) along with numerical integration for obtaining the flow ϕ_x. Newton's method requires the Jacobian

$$J = \frac{\partial r}{\partial x_0} + \frac{\partial r}{\partial \phi_x} x_{x_0}(t_f), \qquad (17)$$

which is dependent upon the trajectory sensitivities evaluated at t_f.

Boundary value problems *per se* are uncommon in power systems. However an application of increasing importance is the calculation of limit cycles (sustained oscillations). Oscillations have been observed in a variety of power systems, from generation [29] to distribution. This latter case is illustrated in Figure 4, where the oscillations were driven by interactions between transformer tapping and capacitor switching.

To solve for limit cycles, (16) can be written

$$x_0 - \phi_x(x_0, T) = 0$$

where x_0 lies on the limit cycle and T is its period. Solution of this boundary value problem via a shooting method requires $x_{x_0}(T)$, which is exactly the Monodromy matrix [27], [30]. The eigenvalues of this matrix determine the stability of the limit cycle.

D. Border-collision bifurcations

When a system trajectory encounters the operating characteristic of a protection device, a trip signal is sent to circuit breakers. If the trajectory almost touches the operating characteristic but just misses, no trip signal is issued. The bounding (separating) case corresponds to the trajectory grazing, i.e., just touching, the operating characteristic but not crossing it. This is a form of global bifurcation; it separates two cases that have significantly different outcomes. Numerous names exist for this phenomena, including C-bifurcation, switching-time bifurcation and border-collision bifurcation.

Examples of such bifurcations can be found in many application areas. They are particularly important in power electronic circuits, where zero-crossings are fundamental to control strategies, and to the switching of self-commutating devices [31]. In fact it has been shown that border-collision bifurcations can provide a path to chaos in simple DC-DC converters [32].

Identifying the critical values of parameters that correspond to a border-collision bifurcation is an inverse problem. Let the operating/switching characteristic be described by $b(x, y) = 0$. A trajectory will be tangential to that characteristic at the point $[x^* \ y^*]^t = \phi(x_0^*, t^*)$ given by

$$b(x^*, y^*) = 0$$
$$(b_x - b_y g_y^{-1} g_x)\big|_{(x^*, y^*)} f(x^*, y^*) = 0.$$

The critical values of parameters are given by x_0^*. This is a special form of boundary value problem. Shooting methods provide the basis for gradient-based algorithms.

E. Critically stable trajectories

Theoretically it's possible to find parameter values that cause a disturbed system to be critically stable, i.e., the trajectory lies exactly on the stability boundary. A perturbation of a parameter in one direction would lead to stability, whereas a perturbation in the opposite direction would cause instability. In practice, the critical values of parameters cannot be determined exactly. However an algorithm that approaches those values has been developed [33].

The algorithm is based on the assumption that the limit set consists only of isolated equilibria. Under that assumption, the stability boundary is composed of the stable manifolds of type-1 unstable equilibrium points (UEPs) that lie on the boundary [34]. Therefore an unstable trajectory must pass through the stable manifold of a type-1 UEP. If a disturbance to the system is critically cleared, the trajectory will lie on the stable manifold of a UEP, and approach that UEP in infinite time. A trajectory that is nearly critically cleared, whether stable or unstable, will pass close by that UEP. But generally the UEP is unknown. Energy function methods require a knowledge of this 'controlling UEP' [35]. The proposed algorithm does not.

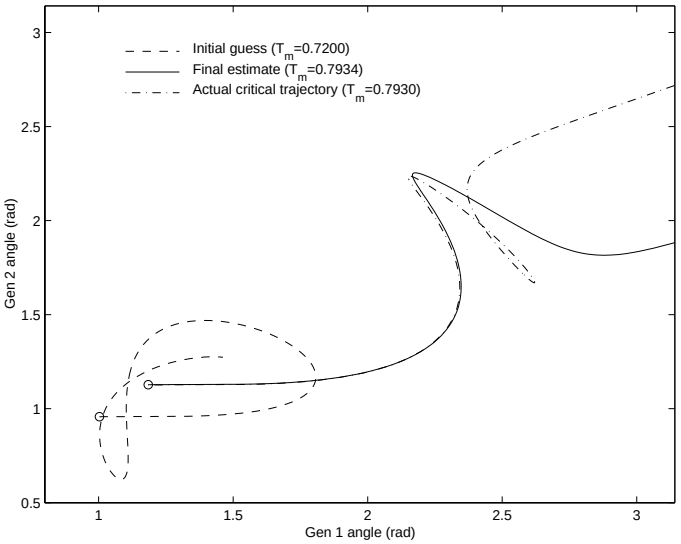

Fig. 5. Convergence to critical trajectory.

Equilibria satisfy

$$f(x, y) = 0$$

along with algebraic constraints (3). Therefore proximity to equilibria can be established using the quadratic cost function

$$\mathcal{J}(x, y) = \frac{1}{2} f(x, y)^t W f(x, y) \tag{18}$$

where W is a diagonal matrix that is used to weight the relative importance of the various elements 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 those 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.

The implementation of this algorithm in [33] uses a Gauss-Newton technique for minimizing $\mathcal{J}$. The gradient information required by the minimization process is provided by trajectory sensitivities. Figure 5 illustrates the process. In that example, the critical value of mechanical torque T_m at one of the generators was obtained. It can be seen from the figure that the actual critical trajectory undergoes convoluted behaviour in the vicinity of the UEP. This introduces local minima into the cost function $\mathcal{J}$, masking the location of the global minimum (exact critical value). However the local and global minima are close, so the error introduced by convergence to the wrong point is small.

F. Optimal control

Optimization problems arise frequently in the analysis of power system dynamics. Examples range from tuning generator AVR/PSSs to determining the optimal location, amount and switching times for load shedding [36]. All these problems can be formulated using a Bolza form of objective function

$$\min_{\theta,t_f} I(x,y,\theta,t_f) \tag{19}$$

where

$$I = \varphi\big(x(t_f),y(t_f),\theta,t_f\big) + \int_{t_0}^{t_f} \psi\big(x(t),y(t),\theta,t\big)dt,$$

θ are the design parameters, i.e., the parameters adjusted to achieve the objective, and t_f is the final time.

The solution of (19) for hybrid systems is complicated by the discontinuous behaviour at events. However these complications largely disappear under the assumption that the order of events does not change as θ and t_f vary, i.e., no border-collision bifurcations occur. This assumption is common throughout the literature, though it is expressed in various ways: transversal crossings of triggering hypersurfaces are assumed in [37], existence of trajectory sensitivities is assumed in [38], and [39] assumes all flows have the same *history*. All statements are equivalent.

Under that assumption, and other mild assumptions, it is concluded in [39] that if I is continuous in its arguments then a solution to (19) exists. Further, [38] shows that if I is a smooth function of its arguments, then it is continuously differentiable with respect to θ and t_f. The minimization can therefore be solved using gradient-based methods. Trajectory sensitivities underlie the gradient information.

If the event ordering assumption is not satisfied, I may be discontinuous. The optimization problem then takes on a combinatorial nature, as each continuous section of I must be searched for a local minimum.

Other optimization problems do not naturally fit the form (19) of the objective function. Cascaded tap-changing transformers provide an interesting example [40]. Minimizing the number of tap change operations is equivalent to minimizing the number of crossings of triggering hypersurfaces. Such a problem, by definition, does not satisfy the earlier assumption requiring constant ordering of events. This minimization is best addressed using switching control design techniques [41], though the solution process is not yet well established.

G. Technical issues

Changes in event ordering, as discussed in the previous subsection, influence all gradient-based algorithms for solving inverse problems. The effect is similar to power flow solution when reactive power limits change status. Algorithms usually converge, though with a slower convergence rate and a reduced region of convergence.

Another interesting aspect of hybrid systems is that trajectories may not be unique in reverse time, even though they are unique in forward time. In other words, the same final value $\phi(x_0,t_f)$ can be reached from different initial values x_0. In such cases, the trajectory sensitivity matrix $x_{x_0}(t_f)$ is singular. This matrix underlies solution algorithms for numerous inverse problems, for example (17). An approach to addressing this issue is to decompose x_0 into components that influence $\phi(x_0,t_f)$ and those that do not. Attention is then restricted to the former group. This is an area of on-going research.

V. Conclusions

The response of power systems to large disturbances often involves interactions between continuous dynamics and discrete events. Such behaviour can be captured by a model that consists of a set of differential-algebraic equations, modified to incorporate impulse (state reset) action and constraint switching (DAIS model).

Models of large systems are most effectively constructed using a modular or object-oriented approach. The desired modularity can be achieved in a practical way with the DAIS model. Components and/or subsystems are modeled autonomously, with connections established via simple algebraic equations. This object-oriented model structure is amenable to symbolic manipulation.

Systematic modeling allows the development of tools for solving inverse problems, including parameter uncertainty and estimation, boundary value problems, locating border-collision bifurcations and critically stable trajectories, and optimal control. The DAIS model is conducive to the efficient generation of trajectory sensitivities. Those sensitivities underlie gradient-based algorithms for inverse problems.

References

[1] L.G. Perez, A.J. Flechsig, and V. Venkatasubramanian, "Modeling the protective system for power system dynamic analysis," *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1963–1973, November 1994.

[2] M.S. Ćalović, "Modeling and analysis of under-load tap-changing transformer control systems," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-103, no. 7, pp. 1909–1915, July 1984.

[3] I. Dobson, "Stability of ideal thyristor and diode switching circuits," *IEEE Transactions on Circuits and Systems I*, vol. 42, no. 9, pp. 517–529, September 1995.

[4] M.D. Lemmon, K.X. He, and I. Markovsky, "Supervisory hybrid systems," *IEEE Control Systems Magazine*, vol. 19, no. 4, pp. 42–55, August 1999.

[5] A. Tarantola, *Inverse Problem Theory*, Elsevier Science, Amsterdam, 1987.

[6] J.F. Hauer, W.A. Mittelstadt, W.H. Litzenberger, C. Clemens, D. Hamai, and P.N. Overholt, "Wide area measurements for real-time control and operation of large electric power systems," DOE Final Report, April 1999.

[7] D.N. Kosterev, C.W. Taylor, and W.A. Mittelstadt, "Model validation for the August 10, 1996 WSCC system outage," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 967–979, August 1999.

[8] I.A. Hiskens, "Nonlinear dynamic model evaluation from disturbance measurements," *IEEE Transactions on Power Systems*, To appear.

[9] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, Springer-Verlag, London, 2000.

[10] C.G. Cassandras and S. Lafortune, *Introduction to Discrete*

Event Systems, Kluwer Academic Publishers, Norwell, MA, 1999.
[11] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.
[12] K.E. Brenan, S.L. Campbell, and L. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1995.
[13] 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.
[14] S. Galán, W.F. Feehery, and P.I. Barton, "Parametric sensitivity functions for hybrid discrete/continuous systems," *Applied Numerical Mathematics*, vol. 31, pp. 17–47, 1999.
[15] P.M. Frank, *Introduction to System Sensitivity Theory*, Academic Press, New York, 1978.
[16] H. Elmqvist, "A structured model language for large continuous systems," PhD Thesis TFRT-1015, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978.
[17] H. Elmqvist, S.E. Mattsson, and M. Otter, "Modelica - A language for physical system modeling, visualization and interaction," in *Proceedings of the IEEE Symposium on Computer-Aided Control System Design*, Hawaii, August 1999.
[18] R. Bacher, "Symbolically assisted numeric computations for power system software development," in *Proceedings of the 13th Power Systems Computation Conference*, Trondheim, Norway, June 1999, pp. 5–16.
[19] I.A. Hiskens and P.J. Sokolowski, "Systematic modeling and symbolically assisted simulation of power systems," *IEEE Transactions on Power Systems*, vol. 16, no. 2, pp. 229–234, May 2001.
[20] F.L. Alvarado and Y. Liu, "General purpose symbolic simulation tools for electric networks," *IEEE Transactions on Power Systems*, vol. 3, no. 2, pp. 689–697, May 1988.
[21] T. Park and P.I. Barton, "State event location in differential-algebraic models," *ACM Transactions on Modeling and Computer Simulation*, vol. 6, no. 2, pp. 137–165, April 1996.
[22] I.A. Hiskens, M.A. Pai, and T.B. Nguyen, "Bounding uncertainty in power system dynamic simulations," in *Proceedings of the IEEE PES Winter Meeting*, Singapore, January 2000.
[23] M.W. Tian and C.-J.R. Shi, "Worst case tolerance analysis of linear analog circuits using sensitivity bands," *IEEE Transactions on Circuits and Systems I*, vol. 47, no. 8, pp. 1138–1145, August 2000.
[24] R.H. Craven, T. George, G.B. Price, P.O. Wright, and I.A. Hiskens, "Validation of dynamic modelling methods against power system response to small and large disturbances," in *Proceedings of CIGRÉ General Session*, Paris, August 1994.
[25] I.A. Hiskens, "Identifiability of hybrid system models," in *Proceedings of the 9th IEEE Conference on Control Applications*, Anchorage, AK, September 2000.
[26] G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins, 2nd edition, 1989.
[27] R. Seydel, *Practical Bifurcation and Stability Analysis*, Springer-Verlag, New York, 2nd edition, 1994.
[28] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer, New York, 1980.
[29] K. Kim, H. Schättler, V. Venkatasubramanian, J. Zaborszky, and P. Hirsch, "Methods for calculating oscillations in large power systems," *IEEE Transactions on Power Systems*, vol. 12, no. 4, pp. 1639–1648, November 1997.
[30] I.A. Hiskens, "Stability of limit cycles in hybrid systems," in *Proceedings of the 34th Hawaii International Conference on System Sciences*, Maui, HI, January 2001.
[31] S. Jalali, I. Dobson, R.H. Lasseter, and G. Venkataramanan, "Switching time bifurcations in a thyristor controlled reactor," *IEEE Transactions on Circuits and Systems I*, vol. 43, no. 3, pp. 209–218, March 1996.
[32] M. di Bernardo, "The complex behavior of switching devices," *IEEE Circuits and Systems Newsletter*, vol. 10, no. 4, pp. 1–13, December 1999.
[33] I.A. Hiskens, M.A. Pai, and P.W. Sauer, "An iterative approach to calculating dynamic ATC," in *Proceedings International Symposium on Bulk Power System Dynamics and Control - IV*, Santorini, Greece, August 1998.
[34] 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.
[35] M.A. Pai, *Energy Function Analysis for Power System Stability*, Kluwer Academic Publishers, Boston, MA, 1989.
[36] C. Moors and T. Van Cutsem, "Determination of optimal load shedding against voltage instability," in *Proceedings of the 13th Power Systems Computation Conference*, Trondheim, Norway, June 1999.
[37] 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.
[38] S. Galán and P.I. Barton, "Dynamic optimization of hybrid systems," *Computers chem. Engng*, vol. 22, Suppl., pp. S183–S190, 1998.
[39] B. Piccoli, "Hybrid systems and optimal control," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, December 1998.
[40] M. Larsson, "Coordinated voltage control in electric power systems," PhD Thesis, Department of Industrial Electrical Engineering and Automation, Lund Institute of Technology, Lund, Sweden, 2000.
[41] R.W. Brockett, "Minimum attention control," in *Proceedings of the 36th Conference on Decision and Control*, 1997, pp. 2628–2632.
[42] 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.
[43] S. Li, L. Petzold, and W. Zhu, "Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem," *Applied Numerical Mathematics*, vol. 32, pp. 161–174, 2000.
[44] D. Chaniotis, M.A. Pai, and I.A. Hiskens, "Sensitivity analysis of differential-algebraic systems using the GMRES method - Application to power systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Sydney, Australia, May 2001.

Appendix

I. Trajectory Sensitivity Equations

Away from events, where system dynamics evolve smoothly, the sensitivities x_{x_0} and y_{x_0} are obtained by differentiating (2)-(3) with respect to x_0. This gives

$$\dot{x}_{x_0} = f_x(t)x_{x_0} + f_y(t)y_{x_0} \quad (20)$$

$$0 = g_x(t)x_{x_0} + g_y(t)y_{x_0} \quad (21)$$

where $f_x \equiv \partial f/\partial x$, and likewise for the other Jacobian matrices. Note that f_x, f_y, g_x, g_y are evaluated along the trajectory, and hence are time-varying matrices. It is shown in Appendix B that the solution of this (potentially high order) DAE system can be obtained as a by-product of solving the original DAE system (2)-(3).

Initial conditions for x_{x_0} are obtained from (8) as

$$x_{x_0}(t_0) = I$$

where I is the identity matrix. Initial conditions for y_{x_0} follow directly from (21),

$$0 = g_x(t_0) + g_y(t_0)y_{x_0}(t_0).$$

Equations (20),(21) describe the evolution of the sensitivities x_{x_0} and y_{x_0} between events. However at an event, the sensitivities are often discontinuous. It is necessary to calculate *jump conditions* describing the step change in x_{x_0} and y_{x_0}. For clarity, consider a single switching/reset

event, so the model (2)-(5) reduces to the form

$$\dot{x} = f(x,y) \tag{22}$$

$$0 = \begin{cases} g^-(x,y) & s(x,y) < 0 \\ g^+(x,y) & s(x,y) > 0 \end{cases} \tag{23}$$

$$x^+ = h(x^-, y^-) \qquad s(x,y) = 0. \tag{24}$$

Let $(x(\tau), y(\tau))$ be the point where the trajectory encounters the hypersurface $s(x,y) = 0$, i.e., the point where an event is triggered. This point is called the *junction point* and τ is the *junction time*. Assume that the trajectory encounters the triggering hypersurface transversally.

Just prior to event triggering, at time τ^-, x and y are given by

$$\begin{aligned} x^- = x(\tau^-) &= \phi_x(x_0, \tau^-) \\ y^- = y^-(\tau^-) &= \phi_y(x_0, \tau^-) \end{aligned}$$

where

$$g^-(x^-, y^-) = 0.$$

Similarly, x^+, y^+ are defined for time τ^+, just after the event has occurred. It is shown in [13] that the jump conditions for the sensitivities x_{x_0} are given by

$$x_{x_0}(\tau^+) = h_x^*\, x_{x_0}(\tau^-) - \left(f^+ - h_x^*\, f^-\right) \tau_{x_0} \tag{25}$$

where

$$\begin{aligned} h_x^* &= \left(h_x - h_y (g_y^-)^{-1} g_x^-\right)\big|_{\tau^-} \\ \tau_{x_0} &= -\frac{\left(s_x - s_y (g_y^-)^{-1} g_x^-\right)\big|_{\tau^-} x_{x_0}(\tau^-)}{\left(s_x - s_y (g_y^-)^{-1} g_x^-\right)\big|_{\tau^-} f^-} \\ f^- &= f(x(\tau^-), y^-(\tau^-)) \\ f^+ &= f(x(\tau^+), y^+(\tau^+)). \end{aligned}$$

The sensitivities y_{x_0} immediately after the event are given by

$$y_{x_0}(\tau^+) = -\left(g_y^+(\tau^+)\right)^{-1} g_x^+(\tau^+) x_{x_0}(\tau^+).$$

Following the event, i.e., for $t > \tau^+$, calculation of the sensitivities proceeds according to (20),(21), until the next event is encountered. The jump conditions provide the initial conditions for the post-event calculations.

II. Efficient trajectory sensitivity computation

Consider the DAIS model (2)-(3) away from events. The trapezoidal approach to numerical integration approximates the differential equations (2) by a set of algebraic difference equations. These algebraic equations are coupled with the original algebraic equations (3) giving

$$x^{k+1} = x^k + \frac{\eta}{2}\left(f(x^k, y^k) + f(x^{k+1}, y^{k+1})\right) \tag{26}$$

$$0 = g(x^{k+1}, y^{k+1}) \tag{27}$$

where the superscripts k, $k+1$ index the time instants t_k, t_{k+1} respectively, and $\eta = t_{k+1} - t_k$ is the integration time step. Equations (26)-(27) describe the evolution of the states x, y from time instant t_k to the next time instant t_{k+1}.

Notice that (26)-(27) form a set of implicit nonlinear algebraic equations. To solve for x^{k+1}, y^{k+1} given x^k, y^k requires the use of a nonlinear equation solver. Newton-based iterative techniques are commonly used. The solution process involves forming and factorizing the Jacobian

$$F_{\mathscr{x}} = \begin{bmatrix} \frac{\eta}{2} f_x - I & \frac{\eta}{2} f_y \\ g_x & g_y \end{bmatrix}. \tag{28}$$

Now consider the sensitivity equations (20)-(21). Using trapezoidal integration, they are approximated by

$$\begin{aligned} x_{x_0}^{k+1} &= x_{x_0}^k + \frac{\eta}{2}\left(f_x^k\, x_{x_0}^k + f_y^k\, y_{x_0}^k + f_x^{k+1} x_{x_0}^{k+1} + f_y^{k+1} y_{x_0}^{k+1}\right) \\ 0 &= g_x^{k+1} x_{x_0}^{k+1} + g_y^{k+1} y_{x_0}^{k+1}. \end{aligned}$$

Rearranging gives

$$\begin{bmatrix} \frac{\eta}{2} f_x^{k+1} - I & \frac{\eta}{2} f_y^{k+1} \\ g_x^{k+1} & g_y^{k+1} \end{bmatrix} \begin{bmatrix} x_{x_0}^{k+1} \\ y_{x_0}^{k+1} \end{bmatrix} = \begin{bmatrix} -\frac{\eta}{2}\left(f_x^k\, x_{x_0}^k + f_y^k\, y_{x_0}^k\right) - x_{x_0}^k \\ 0 \end{bmatrix}. \tag{29}$$

Therefore $x_{x_0}^{k+1}$ and $y_{x_0}^{k+1}$ are obtained as the solution of a linear matrix equation. But notice that the matrix which is inverted in solving (29) is exactly the Jacobian (28) used in solving for x^{k+1} and y^{k+1}. Because that matrix has already been built and factorized to calculate x^{k+1} and y^{k+1}, the solution of (29) involves little extra computation.

To improve simulation speed, (26)-(27) are often solved using a quasi-Newton method. As a result, the factors of $F_{\mathscr{x}}$ may not be available for solving (29) directly. However a number of computationally efficient techniques have been proposed in [42], [43], [44].