Raiden. Power System Transient Stability Simulation Software Prototype

Eugene Mashalov, Joint Stock Company "Scientific and Technical Center of Unified Power System", Ekaterinburg, Russia

***Abstract*—This paper discusses implementation details of the transient stability analysis prototype software based on the implicit integration scheme. The main advantages of an implicit integration scheme with separate treatment of differential and algebraic state variables as well as an integration process with discrete event handling are considered. The results of test simulations and their comparison with existing software packages are presented.**

***Index Terms*—Transient Stability Analysis, Implicit Integration, Discontinuous DAE**

I. Introduction

TRANSIENT stability simulation is one of the most complicated tasks of power system analysis. Besides methodological complexity, it requires intense computation with significant time consumption. Parallel implementation of simulation algorithms is difficult due to the sequential nature of the solution method, which is based on integration of the differential-algebraic system of equations (DAE), which is hardly separable into weakly connected parts. Thus, even modern computer systems do not provide the opportunity for real-time simulation, unless model details are reduced. Transient stability is dominantly used in offline applications, such as facility development, and rarely used for power system control on a time scale close to real-time.

However, some near-real-time implementations have been introduced in the last decade. One of the most successful is the stability monitoring system, used by the System Operator of Russia. The system evaluates the safe limits of network load over a period of 40–900 s with respect to static and transient stability. However, some near-real-time implementations have been introduced in the last decade. One of the most successful is the stability monitoring system, used by the System Operator of Russia. The system evaluates the safe margins of network load over a period of 40–900 s with respect to static and transient stability. These margins are used for system automatic or manual control. The duration of the evaluation cycle is mostly determined by the duration of the transient stability simulation. The aim of reducing the computation time of simulation led to the development of the software prototype based on the implicit integration scheme, instead of the former implementation based on explicit methods. The new algorithm consists of three components of integration: the method, the scheme, and the process. However, some near-real-time implementations have been introduced in the last decade. One of the most successful is the stability monitoring system, used by the System Operator of Russia. The system evaluates the safe margins of network load over a period of 40–900 s with respect to static and transient stability. These margins are used for system automatic or manual control. The duration of the evaluation cycle is mostly determined by the duration of the transient stability simulation. The aim of reducing the computation time of simulation led to the development of the software prototype based on the implicit integration scheme, instead of the former implementation based on explicit methods. The new algorithm consists of three components of integration: the method, the scheme, and the process. The integration method defines the numerical properties of the problem solution. The integration scheme defines the computation process of the method to ensure optimal utilization of the problem properties. And finally, the integration process arranges the operation of the integration scheme, considering discrete events and discontinuities in the model.

II. Transient stability simulation algorithm development

Time spent on transient simulation is proportional to the number of numerical integration steps of the system (1):

$$\begin{cases} \dot{y} &= f(x(t), y(t), t) \\ 0 &= g(x(t), t(t), t) \end{cases} \qquad (1)$$
$$x(t_0) = x_0, y(t_0) = y_0$$

where g and f are smooth vector functions. The time of simulation initialization and computation of the initial conditions is much smaller than the time of the (1) solution. The latter depends on the integration method properties and the system dimension N_e. As a rule, solution time $T_{hn} = O(N_e^3)$, since the integration step in one form or another requires the solution of a system of nonlinear algebraic equations. The value of the integration step h_n is determined by the properties of the DAE. The crucial property of (1) with respect to h_n is the stiffness of the DAE, estimated as the ratio of the maximum and minimum values of the real part of the eigenvalues of its linearized matrix.

A. The integration method

In most cases, the transient stability problem (1) is considered stiff, which encourages using implicit integration methods for its solution. The integration method is a vector function $F(x, y)$ which maps values from the previous step y_{n-1} to current step values y_n. Of course, the current step values of

$x_n \leftarrow x_{n-1}$ are also required, but there are several approaches to performing this. The general form of the implicit method for differential variables is:

$$F(y_n, y_{n-1}, t) = 0 \tag{2}$$

The solution of (3) involves the solution of the nonlinear system of equations, which is associated with some technical complications. That was the main reason for the former implementation of transient stability software, which was developed in the 1980s relying on explicit methods. The solution of (2) implies the solution of the nonlinear system of equations, which is associated with some technical complications. Integration step control bounds solution to:

$$\begin{gathered} ||z(t_n) - z_n(t_n)|| <= \epsilon, \\ z_n(t_n) = [y_n(t_n), x_n(t_n)]^T \\ z(t) = [y(t), x(t)]^T \end{gathered} \tag{3}$$

where

$z_n(t_n)$ is the approximated solution;
$z(t)$ is the exact solution;
ϵ is the error tolerance;

The step control allows one to vary $h_n = t_n - t_{n-1}$ according to (3). For transient phases with rapid changes, the step is reduced. When the transient decays, the step can be increased, which significantly reduces the number of steps required to solve (1).

The implicit method implementation is much more complicated than the explicit one, as it requires a non-linear system of equations solution. In most cases, this solution is provided by the Newton method. Power system equipment models should be implemented with respect to this requirement and provide the ability to construct blocks of partial derivatives of the Jacobi matrix. The use of the automatic implementation system can overcome this problem [1]. Another problem that arises from nonlinearities is the possibility of multiple solutions at the time instants of discontinuities. However, the problems where (1) has discontinuous f and g are well known and there are various approaches developed. Some use rigorous discontinuity modeling [2], while others use integration restart techniques [3].

B. The integration scheme

The DAE (1) solution implies discretization into a pure algebraic system of equations using a chosen integration method. Since the resulting system is nonlinear, an iterative solution method is used. The simplest method is fixed point iteration. This method is usable, but, in many cases, does not provide convergence due to its limitations. The application of the Newton method requires Jacobian matrix formation, which greatly increases the equipment models development complexity. In addition to the system of equations for each model, it is also required to form a block of the Jacobi matrix. However, Newton's method makes it possible to reliably obtain a solution, has good convergence, and, moreover, can be used not only to solve (1) but also to determine the initial conditions at $t = 0$ and at the time points of discrete changes t_d. The weak side of Newton's method is its sensitivity to the quality of the initial guess. This problem is partly solved by using the initial estimation from the previous integration step. An additional factor that improves the convergence of Newton's method can be the scheme of the integration method. A well-known integration approach is the predictor-corrector scheme. With values on the available solution z_{n-1} at the time t_{n-1}, the method can predict the values of z_n using extrapolation. The resulting prediction is used as an initial estimation for solving the equivalent (1) system, in which the differential equations are discretized in a form that depends on the chosen integration method. During the solution, the prediction is corrected so that the values satisfy the given system of equations. This scheme is effectively implemented using multi-step integration methods, which imply saving the data of several completed steps to perform the next step. To solve DAE, this approach is implemented in the Gear method [4]. This method involves the use of the BDF integration method, both for differential and algebraic equations, and provides a simultaneous solution to (1). Such an integration scheme is fully implicit and preserves integration method stability properties.

A variant of the Gear method is implemented in the well-known software EUROSTAG [5]. Since the methods of the BDF family have the property of damping high-frequency oscillations with an increase in the integration step, the Gear method exhibits "hyperstability" and erroneously treats unstable cases as stable. The proposed modification, which consists of applying the Adams method for differential variables, eliminates this drawback. A variant of the Gear method is implemented in the well-known software Eurostag [8]. Since the methods of the BDF family have the property of damping high-frequency oscillations with an increase in the integration step, the Gear method exhibits hyperstability and erroneously treats unstable cases as stable. The proposed modification, which consists of applying the Adams method to differential variables, eliminates this drawback since the 2nd order Adams method is A-stable. The use of the BDF for algebraic variables makes it possible to maintain a higher integration step compared to the Adams method with the same overall stability properties of the integration scheme.

All linear multistep methods of order q, despite significant differences in their properties, can be expressed in Nordsieck form [6]. That vector has a dimension of $1 \times q + 1$. Since the integration method must be A-stable order is constrained $(q <= 2)$ and the maximum dimension of the Nordsieck vector is fixed at 1×3.

At the n-th integration step, Nordsieck vector components of differential and algebraic variables are:

$$Y_n = \left[y_n, h_n\dot{y}_n, \tfrac{h_n^2}{2}\ddot{y}\right], \quad X_n = \left[x_n, h_n\dot{x}_n, \tfrac{h_n^2}{2}\ddot{x}\right] \tag{4}$$

The Nordsieck vector is merely a form to express the Taylor's expansion of x_n and y_n at t_n. If this vector for t_{n-1} is known, the prediction at t_n is:

$$Y_n^0 = Y_{n-1}A, \quad X_n^0 = X_{n-1}A \tag{5}$$

where

Y_n^0 is the differential variables vector extrapolated to t_n;

X_n^0 is the algebraic variables vector extrapolated to t_n;
A is the lower triangular Pascal matrix $q+1 \times q+1$.

The corrected Nordsieck vector at t_n is:

$$Y_n = Y_n^0 + l^Y e_{yn}, \quad X_n = X_n^0 + l^X e_{xn}, \tag{6}$$

where

l^Y is the row vector of integration method for differential variables;
l^X is the row vector of integration method for algebraic variables;
e_{yn}, e_{xn} are the error vectors with respect to the predicted vectors;

The vectors e_{yn}, e_{xn} are normalized:

$$l_1^Y = l_1^X = 1 \tag{7}$$

The corrector equations (6) can be written by components:

$$\begin{array}{rcl} y_n & = & y_n^0 + l_0^Y e_{yn} \\ h_n \dot{y}_n & = & h_n \dot{y}_n^0 + l_1^Y e_{yn} \\ \frac{h_n^2}{2} \ddot{y}_n & = & \frac{h_n^2}{2} \ddot{y}_n^0 + l_2^Y e_{yn} \\ x_n & = & x_n^0 + l_0^X e_{xn} \\ h_n \dot{x}_n & = & h_n \dot{x}_n^0 + l_1^X e_{xn} \\ \frac{h_n^2}{2} \ddot{x}_n & = & \frac{h_n^2}{2} \ddot{x}_n^0 + l_2^X e_{xn} \end{array} \tag{8}$$

From the $\dot{y}_n$ equation from (8):

$$\begin{array}{l} h_n \dot{y}_n = h_n f(x_n, y_n, t_n) \\ h_n \dot{y}_n^0 + l_1^Y e_{yn} - h_n f(x_n, y_n, t_n) = 0 \end{array} \tag{9}$$

Using (7) define:

$$\begin{array}{c} F_Y(e_{xn}, e_{yn}, t) = h_n \dot{y}_n^0 + e_{yn} - h_n f(x_n, y_n, t_n) \\ F_X(e_{xn}, e_{yn}, t) = g(x_n, y_n^0 + l_0^Y e_{yn}, t_n) \end{array} \tag{10}$$

from which the following system can be obtained:

$$\left\{ \begin{array}{rcl} F_Y(e_{xn}, e_{yn}, t) & = & 0 \\ F_X(e_{xn}, e_{yn}, t) & = & 0 \end{array} \right. \tag{11}$$

with Jacobi matrix (n step index is omitted for simplicity):

$$J = \begin{bmatrix} \frac{\partial F_Y(e_x, e_y, t)}{\partial e_y} + \frac{\partial F_Y(e_x, e_y, t)}{\partial e_x} \\ \frac{\partial F_X(e_x, e_y, t)}{\partial e_y} + \frac{\partial F_X(e_x, e_y, t)}{\partial e_x} \end{bmatrix} \tag{12}$$

The combination of (10) and (12) gives:

$$J = \begin{bmatrix} I - h_n l_0^Y \frac{\partial f(x,y,t)}{\partial y} - h_n l_0^X \frac{\partial f(x,y,t)}{\partial x} \\ l_0^Y \frac{\partial g(x,y,t)}{\partial y} + l_0^X \frac{\partial g(x,y,t)}{\partial x} \end{bmatrix} \tag{13}$$

Note that the Jacobi matrix does not become singular with $h_n \to 0$, which allows it to be used to solve DAE for initial conditions y_d, x_d at instants of discontinuities t_d. In addition, step reduction does not cause numerical problems, which, for example, the integration scheme [7] is subject to.

The system of equations (11) can be solved iteratively. Variables at iteration m are:

$$\left\{ \begin{array}{c} y_n^m = y_n^0 + l_0^Y e_{yn}^m \\ x_n^m = x_n^0 + l_0^X e_{xn}^m \\ e_{xn}^0 = e_{yn}^0 = 0 \end{array} \right. \tag{14}$$

The iterative process of solving (11) with respect to e_x and e_y is performed according to the recursive formula (n step index is omitted for simplicity):

$$\begin{bmatrix} e_y^{m+1} \\ e_x^{m+1} \end{bmatrix} + \begin{bmatrix} e_y^m \\ e_x^m \end{bmatrix} + (J^m)^{-1} + B^m \tag{15}$$

where:

$$B^m = \begin{bmatrix} h_n f(x^m, y^m, t) - h\dot{y}^0 - e_y^m \\ -g(x^m, y^m, t_n) \end{bmatrix} \tag{16}$$

and:

$$J^m = \begin{bmatrix} I - h_n l_0^Y \frac{\partial f(x^m, y^m, t)}{\partial y} - h_n l_0^X \frac{\partial f(x^m, y^m, t)}{\partial x} \\ l_0^Y \frac{\partial g(x^m, y^m, t)}{\partial y} + l_0^X \frac{\partial g(x^m, y^m, t)}{\partial x} \end{bmatrix} \tag{17}$$

When the process (15) converges, the Nordsiek vectors X_n and Y_n at the t_n are updated using (6).

C. Local truncation error control

The solution of (15) is the difference between predicted and corrected values of the state variables. It is used for local truncation error estimation. The next integration step and integration method optimal order can be further determined using error estimation. Local truncation error d_n at the step n:

$$d_n = C_{q+1} q! l_q e_n \tag{18}$$

where C_{q+1} is the integration method constant.

Since state variables may have very different magnitudes, it is common to use weighted differences for error estimation:

$$W_{in} = Rtol_i |Z_{i,n-1}| + Atol_i \tag{19}$$

where:

$Atol_i$ is the absolute tolerance of i-th state variable;
$Rtol_i$ is the relative tolerance of i-th state variable.

The condition (20) is used to accept step n.

$$||d_n|| = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{d_{i,n}}{W_{i,n}} \right)^2} \leq 1 \tag{20}$$

where N is the number of state variables. If (20) holds, the step and the order of the integration method are adjusted for the next step. The step is rejected and repeated otherwise with a smaller h.

The choice of the norm type for d estimation has a strong influence on the quality of the simulation. The default option is l^2-norm, which is acceptable both for the quality of the solution and step size control. As the (1) is stiff, differential equations with small time constants tend to produce large differences between predicted and corrected values. The l^2-norm averages these differences and allows step size to be larger. However, this averaging property degrades the quality of simulation for models with large dimensions due to the

"masking" of unacceptable errors in some state variables. In some cases, these errors have a crucial influence on the stability simulations and may lead to erroneous estimation of unstable cases as stable. Using l^∞-norm overcomes this problem but leads to an unacceptable time step decrease. A hybrid norm was proposed in [8], which combines properties of l^2- and l^∞-norms. In this paper, a similar approach based on the weighting of norms is applied.

The operations of integration step and order change induce extra computation costs and should not be used at every step. Step size and order control are based on the strategy described in [9], where these aspects are considered in relation to the pure differential system of equations. Since the problem being solved is differential-algebraic and stiff, additional measures for convergence control of the corrector and excessive step change suppression are implemented on the basis of [7].

D. Adams integration method ringing suppression

The implicit 2nd-order Adams method, also known as the trapezoidal rule, has the property of "ringing" with the increase of the integration step. Ringing appears as oscillations of state variables at transient decay phases where the integration step is relatively large. Ringing does not affect the quality of simulations considerably, but prevents the growth of the integration step, despite the fact that the physical process tends to a steady state.

The 2nd-order Adams method (21) applied to the model equation $\dot{y} = \lambda y, y(0) = y_0$ gives expression (22).

$$y_n = y_{n-1} + \frac{h}{2}\left(\dot{y}_{n-1} + \dot{y}_n\right) \tag{21}$$

$$y_n = y_0 \left(\frac{2+h\lambda}{2-h\lambda}\right)^n \tag{22}$$

$$\lim_{h\to\infty}\left(\frac{2+h\lambda}{2-h\lambda}\right) = -1 \tag{23}$$

As the integration step increase the (23) holds, which leads to the state variable sign change $(-1)^n$ at every integration step. To suppress ringing, it is necessary to eliminate the influence of the property (23) to the solution. There are several approaches to this problem. For the chosen integration scheme, good results are obtained by replacing $\dot{y}_n$ proposed in [10], after obtaining the solution y_n by (21), by the value $\dot{y}_n^{BDF}$, computed by (24).

$$\dot{y}_n^{BDF} = \frac{\alpha^2 y_{n-2} - (1+\alpha)^2 y_{n-1} + (1+2\alpha) y_n}{h_n(1+\alpha)}$$
$$\alpha = \frac{h_n}{h_{n-1}} \tag{24}$$

At the step $n+1$ the expression (21) would use $\dot{y}_n^{BDF}$ instead of the value $\dot{y}_n$ computed at the step n, as shown in (25).

$$y_{n+1} = y_n + \frac{h}{2}\left(\dot{y}_n^{BDF} + \dot{y}_{n+1}\right) \tag{25}$$

Since replacement $\dot{y}_n^{BDF} \to \dot{y}_n$ changes integration method stability properties, it is applied only when ringing is detected. The extra conditions to apply this artificial damping are $h_n > 0.1$ s and a cool-down period of several steps before the next damping. The effect of damping is shown in Fig. 1 and Fig. 2.

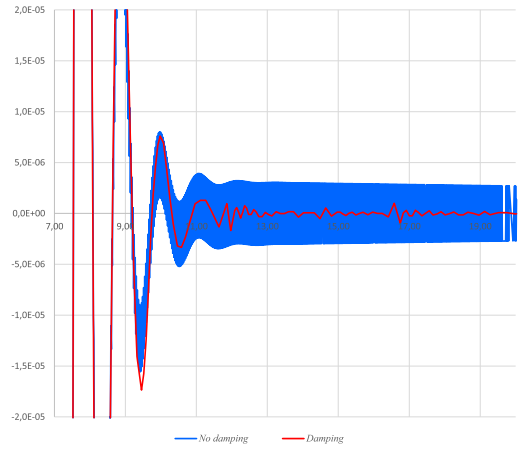

Fig. 1. The ringing suppression effect of 2nd-order Adams damping on state variable evolutions

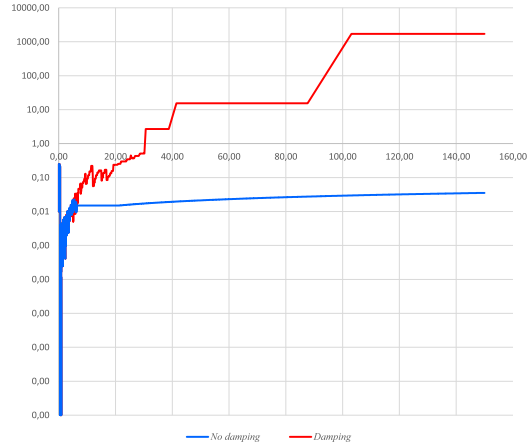

Fig. 2. The ringing suppression effect of 2nd-order Adams damping on integration step size evolutions (log scale)

E. The integration process

The integration method and scheme considered above assume that the (1) is continuous. When simulating practical transients, this condition is not met since some of the f and g are not smooth over their entire domain due to constraints or are discrete. In addition, to simulate discrete events, some of the equations of the system have to be modified. In some cases, the dimension of the system also changes. Problems in which system (1) is not continuous are the separate problem class for DSAR – Differential Switched Algebraic and State Reset equations [11]. An obvious approach for solving such problems is integrating a sequence of continuous DAEs.

The problem of DSAR treatment consists of the arrangement of a transition from one continuous DAE to the next one at a time instant defined with consistent initial conditions. The last problem is not trivial, since the nonlinearity and the presence of discrete functions make multiple solutions possible. Due to the additional information available in the history of solution and specified in the form of rules in the

models [1], the ambiguity can be resolved by a formal choice of the most plausible combination of state variables.

Thus, a full-fledged algorithm for tranient stability simulation should implement the integration process, which governs the integration scheme and solves the following problems:

1) Discrete events management.
2) Discontinuities handling.
3) Calculation of initial conditions and restart of integration
4) State events time instants location.

The discrete event is a command to modify (1) at an arbitrary time instant t_d. Command execution requires at least a restart of the integration. In general, some modifications of (1) structure are also needed. The new consistent initial conditions at t_d must be calculated to resume integration.

Discrete events can be divided into two groups: time events and state events [3]. The events of the first group occur at the moments of time that are known before simulation. These include simulation stop time, application of disturbances or triggered time delays. State events occur when some conditions are met. They are connected with threshold elements, limiters, comparators, relays and so on. Their time of occurrence must be determined during the simulation.

REFERENCES

[1] E. Mashalov, "Automatic implementation of equipment models for transient stability simulation by symbolic manipulations," *Researchgate*, 2022. [Online]. Available: https://rgdoi.net/10.13140/RG.2.2.29110.78404

[2] A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides*, F. M. Arscott, Ed. Springer Netherlands, 1988. [Online]. Available: https://doi.org/10.1007/978-94-015-7793-9

[3] F. E. Cellier and E. Kofman, *Continuous System Simulation*. Springer, New York, 2006.

[4] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, 1971.

[5] J. Astic, A. Bihain, and M. Jerosolimski, "The mixed Adams-BDF variable step size algorithm to simulate transient and long term phenomena in power systems," *IEEE Transactions on Power Systems*, vol. 9, no. 2, pp. 929–935, 1994.

[6] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*. Springer Berlin, Heidelberg, 1993.

[7] P. Linda, "A Description of DASSL: A Differential/Algebraic System Solver, SAND82-8637," Sandia Labs, Tech. Rep., 1982.

[8] "Deliverable d4.1 of the pegase. algorithmic requirements for simulation of large network extreme scenarios," CRSA, RTE, TE, TU/e, Tech. Rep., 2011. [Online]. Available: http://fp7-pegase.com

[9] K. Radhakrishnan and A. C. Hindmarsh, "Description and use of LSODE, the livemore solver for ordinary differential equations," Office of Scientific and Technical Information (OSTI), Tech. Rep., Dec. 1993. [Online]. Available: https://doi.org/10.2172/15013302

[10] J. A. Lee, J. Nam, and M. Pasquali, "A new stabilization of adaptive step trapezoid rule based on finite difference interrupts," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. A725–A746, Jan. 2015. [Online]. Available: https://doi.org/10.1137/140966915

[11] I. Hiskens and P. Sokolowski, "Systematic modeling and symbolically assisted simulation of power systems," *IEEE Transactions on Power Systems*, vol. 16, no. 2, pp. 229–234, 2001.

III. BIOGRAPHY SECTION

Eugene Mashalov Graduated from Electrotechnical Faculty, Ekaterinburg, Urals State Polytechnical University, 1997. Received Ph.D degree in 2000 from the same university. Works for JSC "Scientific and Technical Center of Unified Power System", Ekaterinburg, Russia.
mashalov@gmail.com
www.inorxl.com