OpenDSS Dynamics Mode

June 22, 2011

This document describes how the *Dynamics* mode solution in the OpenDSS program works. Examples are provided from the Generator element, which is currently the most common element used in simulations in Dynamincs mode.

The Basic Algorithm

Each time step in the dynamics mode solution executes the following steps (extracted directly from the code in SolutionAlgs.Pas):

```
      Increment_time;

    {Predictor}
      IterationFlag := 0;
      IntegratePCStates;
      SolveSnap;

    {Corrector}
      IterationFlag := 1;
      IntegratePCStates;
      SolveSnap;
```

The algorithm is currently a simple predictor-corrector method with one step of correction. The *IterationFlag* variable indicates to the integration routines whether the solution is in the predictor step or the corrector step.

The dynamics algorithm was first implemented with no corrector step, but after some experimentation it was found that the machine electromechanical transients behaved much better with one step of correction.

The *IntegratePCStates* function cycles through each Power Conversion (PC) element in the circuit and executes its *IntegrateStates* function. *IntegrateStates* is a *virtual* function that is overridden in models that actually have something to integrate. For example, Generator elements have state variables to integrate. However, others do not and this function is simply empty.

Only PC elements have states that are integrated. Power Delivery (PD) elements are constant impedance elements simply defined by a primitive Y matrix.

Going into Dynamics Mode

The first requirement is to get a solution to the base power flow to initialize the dynamics solution. This is to initialize the state variables of the various dynamic elements to approximately match the power flow. If the power flow will not converge, you can do a "direct" solution, which is a non-iterative solution with PC elements (Loads, etc) modeled as constant impedances. This is generally good enough to initialize the voltages at the generator terminals.

In addition to *Dynamics* mode, the OpenDSS program goes into dynamics solution mode for *FaultStudy* and *MonteFault* (MF) solution modes so that contributions from Generator objects and other active objects are more accurately captured.

The steps the program executes when going into Dynamics mode from one of the power flow solution modes is as follows:

- Initialize state variables in all PC Elements
    - For example, in a Generator object currently:
        - Compute voltage, E_1, behind Xd' to match power flow (approximately)
        - Initialize the phase angle, θ, (power angle) of equivalent voltage source
- Set derivatives of the state variables to zero
    - For the Generator,
        - Speed (relative to synch frequency)
        - Angle
- Set controlmode=time
    - When running in time steps of a few seconds or less, controls that depend on the control queue for instructions on delayed actions will be automatically sequenced when the solution time reaches the designated time for an action to occur.
- Set a flag to preserve Node voltages in case Y matrix changes.
    - Y matrix changes could occur if a switch were to open during the simulation, which might re-order the bus lists. This flag prevents loss of data by reassigning the Node voltage after the re-ordering.

Integrating the State Variables

Integration is performed within the PC element modules themselves, not in the Solution module. Therefore, it is possible to have different integration formulae for different classes of devices.

As of this writing, the Generator employs a simple trapezoidal-rule predictor-corrector integration method. This will be described in the next section.

Predictor and Corrector Steps

The predictor step offers algorithm designers the opportunity to insert a predictor formula at the beginning of the process. It may differ from the corrector step only in that it updates the initial value of the state variables for this time step based on derivatives at previous solution. The following three steps are performed for both the predictor and corrector steps:

1. Compute derivatives for this time step
2. Integrate state variables
3. Solve the circuit with this guess

For the present implementation of the Generator object, the state variables are the speed (relative to synchronous speed) and the shaft angle, θ. The derivatives of these are computed and then integrated in a trapezoidal formula as follows:

Derivative Calculation:

$$\frac{dv}{dt} = \frac{Pshaft - Pterm - Dv}{M}$$

$$\frac{d\theta}{dt} = v$$

Where,

v = shaft speed (velocity) relative to synchronous speed

θ = shaft, or power, angle (relative to synchronous reference frame)

Pshaft = shaft power

Pterm = terminal power out

D = damping constant

M = mass

Integration

Trapezoidal integration formula for θ, for example:

$$\theta_{n+1} = \theta_n + \frac{\Delta t}{2}\left[\left.\frac{d\theta}{dt}\right|_n + \left.\frac{d\theta}{dt}\right|_{n+1}\right]$$

Δt = time step size

In the Predictor step, the integration routine captures the state variables and their derivatives from the previous time step (moves the *(n+1)* values to the *n* values in the formulae above). Then it makes a Predictor calculation of the derivatives at the new step based on the present values of the state variables.

Based on the results of the Predictor step, a new guess at the derivatives at the present *(n+1)* step is computed in the Corrector step. Then the guess at the state variables is corrected.

Only one Corrector step is executed. A circuit solution is performed between the predictor and corrector steps and after the corrector step.

Circuit Solution in Dynamics Mode

The circuit solution proceeds more or less like it would for a normal power flow solution. The Load elements are presently treated the same since load dynamics have not been

implemented. They are sinks or sources of power, depending on sign. However, the Generator, and any other elements with dynamics models, will behave differently. The present implementation of the Generator model computes its terminal currents using the voltage source, adjusted for the new phase angle computed by the integration steps, rather than a specified power and power factor as it is modeled in a power flow solution. Thus, the power from the Generator will depend on system conditions with might be the result of a disturbance (that's usually why you would be simulating dynamics).

As a point of reference, in *Harmonics* mode, the *Loadmodel* option is set to *Admittance* and a direct solution is performed. This is a non-iterative solution mode. However, in Dynamics mode, the load flow is normally solved iteratively for each guess at the Generator voltages.

Computing Terminal Currents in Dynamics Mode (Generator)

As an example of computing terminal currents, we will examine the default Generator object algorithm since it is the most commonly-used at present. The electrical model for a 3-phase Generator object is illustrated in Figure 1.

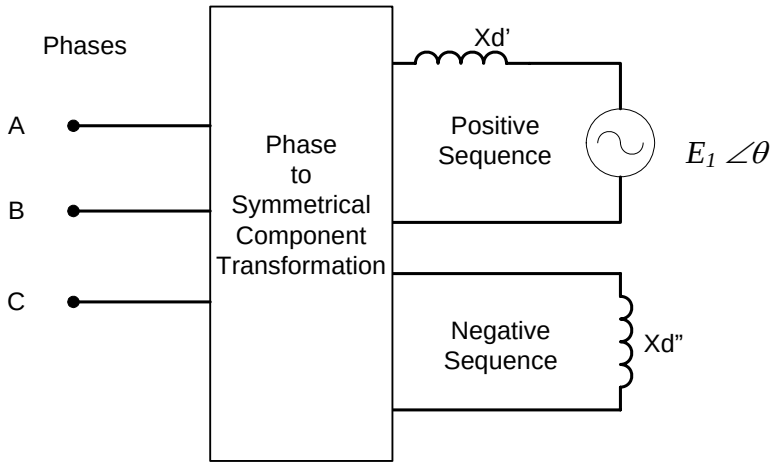

Figure 1. Default Model of Electrical Part of Generator Object in Dynamics Mode

In any of the power flow modes (Snapshot, etc), the Generator object is treated as a power source like most other power conversion elements. Upon entering dynamics mode, the Generator is converted to a positive-sequence voltage source behind a reactance of the form shown in Figure 1. Note that this is a symmetrical component model with the generator part being represented by positive- and negative-sequence networks as shown. The positive-sequence network consists of a voltage source, $E_1 \angle\theta$, behind the value specified for the Xdp property of the Generator model, representing the transient reactance, *Xd'*. E_1 and θ are initialized so that the initial solution in dynamics mode approximates the most recent power flow solution. Thus, simulations will be better if the initializing power flow solution is reasonably well balanced (i.e., no fault conditions – do that *after* entering dynamics mode). The generator excitation is assumed to be only positive sequence.

The negative-sequence network consists of the value specified for the Xdpp property of the Generator. This is usually closer to the actual negative sequence reactance of a machine. The actual negative sequence reactance of the generator many be entered for Xdpp if known.

The terminal currents are computed as follows:

- The phase domain (ABC) voltages are transformed to symmetrical component (012) voltages.
- The symmetrical component currents are computed based on the present value of E_1 and θ.
    - θ is determined in the integration step. In the default model this is based on a simple single-mass model and the difference between the shaft power and the electrical power.
    - In the default model, no exciter or governor action is modeled. Therefore, E_1 and the shaft power remain constant for the duration of the simulation. User-written modules could change these values.
- Finally, the symmetrical component currents are transformed back into the phase domain. The terminal currents are used to compute the *compensation* currents that are used in the circuit solution algorithm (see below).

Compensation Currents

Once the terminal currents have been computed, the *compensation currents* from all PC elements that appear in the system nodal admittance equations in the Normal solution mode are computed (see Figure 2).

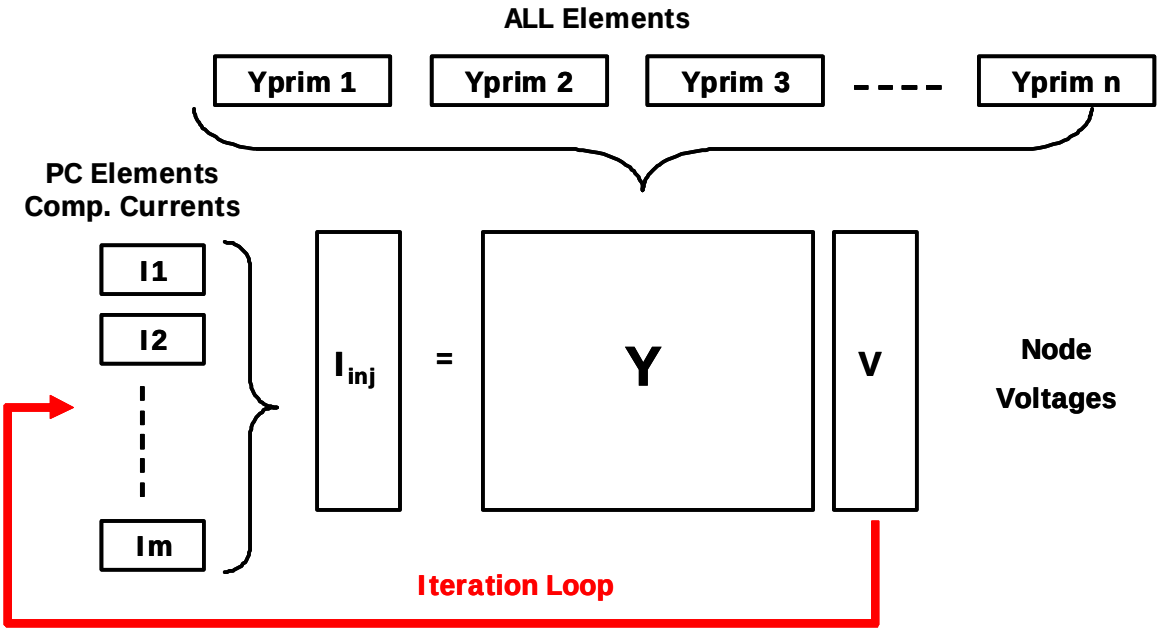

Figure 2. Normal Solution Iteration Loop

These currents are the values of the current source in the Norton equivalent representing the Generator. This is illustrated in Figure 3. The equivalent consists of a constant impedance (actually, admittance) part that is included in the system **Y** matrix in Figure 2.

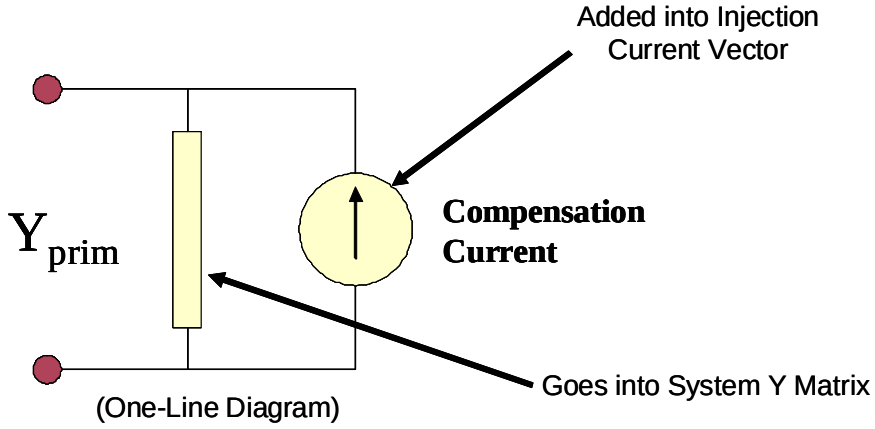

Figure 3. Norton Equivalent Model of Power Conversion (PC) Elements

For the default Generator model, the constant impedance part of the model represented by the Yprim matrix would yield the power at the time of entering Dynamics mode if the terminal voltage were 100% of the specified rated voltage. The compensation current is the difference between the terminal currents computed as described above and the currents in the Yprim branch.

Each element of the multiphase compensation current array is *added* into a particular element of the *Iinj* array shown in Figure 2. The *Iinj* array is a collection of the Norton equivalent current sources in all the PC elements in the circuit.