Achieving Accurate Results with a Circuit Simulator

Kenneth S. Kundert and Ian Clifford
Cadence Design Systems
San Jose, California

Abstract

Techniques for resolving convergence difficulties and controlling error will be presented. The presentation will be geared toward describing how designers can recognize and control common simulation problems.

1 Introduction

Circuit simulation is an important tool that is heavily used when designing analog electronic circuits. Circuit simulators such as SPICE allow new designs to be evaluated quickly and at considerably less expense than the only other alternative, fabrication. However, even though circuit simulators have been heavily used for over 20 years, getting a circuit simulator to converge and give an accurate answer is still considered an art. There is considerable folklore on how to use simulators successfully; however, much of it is based on idiosyncrasies of particular simulators or tricks that are based largely on luck. This paper presents one person's suggestions on getting a simulator to behave, based on 15 years of experience, both as a user and as a developer of circuit simulators.

2 Convergence Difficulties

On each Newton iteration, the circuit is linearized and the solution to this linear circuit is computed and used in the next iteration. If Newton's method starts from a point near the solution, it is guaranteed to find that solution if the component model equations are continuously differentiable and if the solution is isolated. Generally, convergence problems occur due to poor starting points.

As a practical matter, it has been observed that errors in specifying circuit connectivity, component values, or model parameter values will often cause convergence problems. Circuit simulators provide a topology checker that can find many connectivity problems before Newton's method is applied. However, if convergence problems occur, the netlist should be carefully checked both for topological errors and unreasonable model or instance parameters.

2.1 Isolated Solutions

Newton's method requires that the solution be isolated. A solution is isolated if changing it slightly in any direction would cause Kirchhoff's laws to be violated. Many circuits have solutions that are not isolated. If the nonisolated solutions result from a structural property of the circuit, the topology checker will usually identify the problems. In particular, it searches for zero-resistance loops and nodes that do not have paths to ground for DC currents. For example, consider the DC analysis of a circuit containing a subcircuit that is completely isolated from ground except possibly for capacitors. While the node voltages within the subcircuit are well defined with respect to other nodes in the subcircuit, they may all be raised or lowered in unison with respect to ground without violating Kirchhoff's current law. Thus there is an infinite continuum of solutions. Another scenario involves a loop of ideal inductors. Since ideal inductors have zero resistance at DC, they can support a DC current even with no applied voltage. Thus, an arbitrary amount of current may be circulating in the loop without affecting the validity of the solutions.

The topology checker will not find situations that cause nonisolated solutions that depend on component parameter values. A common example of this situation is when a CMOS inverter is constructed with FETs that have their model parameters set such that they have infinite output impedance in saturation. When either the N- or

P-type device is in the ohmic region, the solution is unique, but when both devices are saturated, there exists a finite range of output voltages that all satisfy Kirchhoff's current law. In this situation, the linearized circuit that Newton's method forms on each iteration (the Jacobian) is singular. This is one of several situations that cause problems when using overly simplified models. SPICE tries to avoid this problem by adding a resistor with conductance `gmin` across each junction of every nonlinear component. This generally solves the problem.

2.2 A Good Starting Point

You can specify the starting point to SPICE using the `nodeset` statement. Providing a complete set of node voltages and branch currents is best, but if that is not possible, an incomplete set often helps. If given an incomplete set, SPICE tries to compute the remaining unknowns by performing an initial DC solution with the given voltages and currents forced to their specified values. The `nodeset` statement can be used not only to aid convergence, but also to bias the simulator to finding a particular solution when more than one exist.

2.3 Continuation Methods

Given an initial guess from one of the above methods, SPICE applies Newton's method attempting to find a solution. If it is unsuccessful, you can tell it to use a continuation or homotopy method to find the solution (in SPICE2, set `itl6` nonzero, in other simulators, the continuation methods are automatically used). Continuation methods are slower but more robust than Newton's method. They start by modifying the circuit in such a way that the solution to the modified circuit is known or easy to compute, and such that a parameter controls the amount of modification. Once the solution has been found for the modified circuit, the parameter is slowly changed to the value that will cause the circuit to return to its original form. As the parameter is changed, the solution is computed at each step, using the solution from the previous step as the starting point. As long as the solution changes continuously as a function of the parameter, the previous solution will always be a good starting point and Newton's method will converge.

There are three continuation methods in common use in simulators today, source stepping, `gmin` stepping, and pseudo-transient. SPICE2 provides source stepping, SPICE3 provides `gmin` stepping, ASTAP provides pseudo-transient, and Spectre provides all three.

`gmin` stepping starts by placing small resistors in parallel with all nonlinear devices and computing a solution. The solution is easy to compute because the nonlinear behavior of the devices is swamped out by the resistors. The size of the resistors is slowly increased, and the solution is computed each time the resistors are changed. Eventually the resistors are so large that they no longer affect the circuit. They are then removed completely and the DC solution computed.

Source stepping starts by setting all source voltages and currents to zero, and slowly ramping them to their full value. The solution is recomputed each time the source level is changed, with all but the final solution discarded.

With the pseudo-transient method, a capacitor is installed in parallel with each nonlinear device with a zero initial condition (the capacitors and inductors naturally in the circuit are ignored by setting them to zero). Time (rather, the "pseudo-time") is swept from zero to infinity in an attempt to find the DC solution. The capacitors should cause the solution waveform to be a continuous function of time, the continuation parameter. This continuation method works well as long as adding the capacitors does not convert the circuit into an oscillator.

2.4 Transient Convergence

The transient analysis has similar convergence properties as a continuation method. In fact, it is a continuation method with time being the continuation parameter. If the transient analysis has convergence difficulties at a particular time point, as long as the solution waveforms are continuous, it should always be possible to achieve convergence by taking a smaller time step because eventually as the time step is reduced, the solution at the previous time point will enter the region of convergence for the current time point.

There are two situations for which reducing the time step does not improve convergence. First, if the models for the nonlinear capacitors are not continuous, Newton's method can get hung up on the discontinuities and never converge. Shrinking the time-step actually makes things worse because it results in the discontinuous capacitors dominating over the presumably continuous linear and nonlinear resistors. Unfortunately, discontinuous models are a fault found in circuit simulators too often. The only real solution to this problem is for the simulator to be fixed so that it supplies continuous models.

The second situation for which reducing the time step does not improve convergence is if the waveforms exhibit discontinuous jumps. This can occur in circuits with positive feedback such as latches and Schmitt triggers when they contains nodes that do not have a capacitive path to ground. In this case, shrinking the time step does not bring the previous time point into the region of convergence for the current time point because the transition is infinitely fast. In practice, waveforms generated by circuits cannot jump discontinuously, and so the circuit must be incompletely modeled. Generally, what is needed is a small capacitor from the troublesome node to ground.

2.5 If Your Simulator Fails to Converge

Here are some suggestions to try if you are having convergence problems.

2.5.1 DC Analysis

1. Heed any warnings the simulator gives. This may seem obvious, but I have found that it is not so obvious that people always do it.

2. Carefully check all parameter values to assure they are reasonable

3. Use `nodeset` statements to provide a good starting point for as many nodes as possible.

4. Loosen tolerances, particularly `abstol`. If tolerances are set too tight, they might preclude convergence.

5. Consider increasing `gmin` above its default value of $10^{-12}\mho$. Be careful not to increase it to a degree that it interferes with the proper operation of the circuit.

6. If all else fails, replace the DC analysis with a transient analysis and modify all the independent sources to start at zero and ramp to their DC values. Run the transient analysis well beyond the time when all the sources have reached their final value (remember that transient analysis is very cheap when all of the signals in the circuit are not changing) and use the final solution to generate `nodesets`. To make the transient analysis more efficient, eliminate the local-truncation error criterion by setting the option `lvltim`=1 (this is the *only* time you should use this option).

   Occasionally, this approach will fail or be very slow because the circuit is or contains an oscillator. Try to disable the oscillator before using this method.

7. If you cannot get the simulator to converge when computing the initial point of a transient, skip the initial point computation by specifying `uic` on the transient analysis statement.

Long amplifier chains (inverter chains or ring oscillators) may have convergence problems because their high gain nature causes the matrix package to overflow. Carefully choose an initial guess with `nodeset` statements. If the solution is in a low gain region (such as with an inverter chain), use `nodesets` to initially bias the circuit into the low gain region.

2.5.2 Transient Analysis

There are two strategies used to circumvent convergence problems in the transient analysis: reduce the effect of discontinuities in the nonlinear capacitors and eliminate discontinuous jumps in the solution.

1. When specifying the nonlinear device model parameters, be sure to give a complete capacitance model. Do not use simplified device models that do not have capacitances.

2. Be sure to give the source and drain areas for all MOSFETS. This results in the junction capacitors being modeled. Also, give all overlap capacitances.

3. If, by a process of elimination, you can identify a nonlinear capacitance that seems to have a discontinuity, try to simplify or even just modify the nonlinear capacitor model. This can sometimes eliminate the discontinuity.

4. Consider adding a small linear capacitor from every node to ground.

5. If all else fails, loosen `trtol` or `reltol` and widen transitions in the stimulus waveforms. This can sometimes cause the simulator to just jump past the convergence difficulties.

3 Accuracy

The accuracy of a simulation is dependent on three factors, the accuracy of the models, the error injected by the simulator algorithms, and the circuit itself. If the models do not accurately reflect the physical effects that are important to the circuit behavior, then the answer will be wrong. Similarly, if the model does not match the physical device because the model parameters are poorly chosen, again, the answer will be in error. However, even if the model does a good job of reproducing the important characteristics of the device, the simulator itself can inject error into this solution.

These errors, if not well controlled, can result in significant error. The circuit itself can either magnify errors made by the simulator and the models, or, conversely, it can be very tolerant of such errors. Circuits that can be very sensitive to simulator errors include oscillators and charge storage circuits such as switched-capacitor circuits.

3.1 Model Accuracy

An important prerequisite to an accurate solution is the use of the right model for the particular problem. This mostly relates to MOS models because there are so many of them. Some important issues to consider when choosing a MOSFET model are charge conservation (important in charge storage circuits), output resistance (important with high gain amplifiers), subthreshold conduction (important with low power circuits), impact ionization (important when output resistance is critical and with circuits that are sensitive to substrate current), and the ability to accurately model mobility modulation, short channels, and thin oxides (important with today's small devices). The Shichman-Hodges model (SPICE MOS level-1) is very efficient, though it is generally considered too inaccurate to use. SPICE MOS level-2 is a physically based model that is poor for analog applications or for small devices. It is also expensive to evaluate, and so causes the simulator to run slowly. SPICE MOS level-3 can better handle small devices, but all of these models as implemented in Berkeley SPICE and its descendants do not conserve charge. The BSIM1 model is charge conserving and is good for short channels; however, it is not appropriate for analog circuits because of its poor output conductance modeling. BSIM2 works well for both small device size (good to 0.2μm gate length and 36 angstrom oxide thickness) and analog circuits because of its good output conductance and subthreshold conductance modeling. Both BSIM models are charge conserving. BSIM2 is the only commercially available MOS model suitable for analog design.

It is sometimes possible to accelerate the simulation of circuits by substituting faster models in sections of the circuit that are relatively insensitive to small errors in the models. For example, if one were simulating a mixed analog-digital circuit where the analog portion was very sensitive and required an accurate model, but the digital section did not, then one could use the simpler and faster models to simulate the logic.

With any semiconductor model, you should be careful to avoid very small parasitic resistances. The general rule of thumb is, if the voltage drop across the resistor will not be significant, discard the resistor or reextract the parameters without it and let the other parameters absorb its effect. Eliminating the resistor will make the simulation run faster, and may eliminate convergence or accuracy problems.

There is one common error made when using the bipolar model that could result in accuracy being degraded. The SPICE bipolar model only partially implements the substrate junction. The junction is always reverse-biased in practice so it was felt that it was only necessary to implement the junction capacitance, and not the junction diode. However, if you do not specify the substrate terminal, SPICE connects it to ground, which is often inappropriate and may result in the junction being forward-biased. Since the diode is missing, most designers do not notice the mistake, however the junction capacitance can be 3-5 times larger than it would be if properly biased.

3.2 DC Analysis Accuracy

In a DC or operating point analysis, there are three things that contribute error in the solution. First, the models may not be completely accurate. Second, the simulator may add components to the circuit that the user did not explicitly specify, such as `gmin`. Lastly, the convergence criteria contribute error because they stop the Newton-Raphson iteration before the nonlinear equations are solved exactly.

3.2.1 Gmin

Most simulators add a very small conductance of `gmin` across nonlinear devices to prevent nodes from floating if the nonlinear devices are turned completely off. By default, `gmin`$=10^{-12}$.

The `gmin` conductors affect the accuracy of the solution because they change the circuit being solved by the simulator. Most circuits are tolerant of the small currents that flow through the `gmin` conductors, however some circuits are not. For example, sample-and-hold circuits or other circuits that try to hold charge on a capacitor for a long period of time are sensitive to the small currents that flow through the `gmin` conductors.

3.2.2 Convergence Criteria

Newton's method continues to iterate until the convergence criteria are satisfied. There are two criteria use by SPICE to determine when a circuit has converged,

$$|v_n^{(j)} - v_n^{(j-1)}| < \texttt{vntol}+ \texttt{reltol}\max(|v_n^{(j)}|, |v_n^{(j-1)}|) \tag{1}$$

and

$$|f_k(v^{(j)}) - f_k(v^{(j-1)})| < \texttt{abstol}+ \texttt{reltol}\max(|f_k(v^{(j)})|, |f_k(v^{(j-1)})|) \tag{2}$$

where $v_n^{(j)}$ is the voltage on node n at iteration j and $f_k(v^{(j)})$ is the current through nonlinear device k. By default, `reltol`=0.001, `vntol`=1μV, and `abstol`=1pA. `reltol` is called the relative convergence tolerance because it specifies how small the update must be relative to the node voltage. `reltol` allow you to simulate high voltage circuits and low voltage circuits without adjusting the convergence criteria. `vntol` and `abstol` are referred to as absolute tolerances. They become important when a particular voltage or current are nearly zero. In this case, just using the `reltol` criterion would force the update to be microscopic before convergence was allowed. In some cases the required update would be smaller than the computer round-off error, in which case convergence would never be allowed. `vntol` (`abstol`) prevents these problems from occurring by causing any update smaller that `vntol` (`abstol`) to be accepted.

SPICE considers $v^{(j)}$ a solution if (1) and (2) are both satisfied. Even if both of these criteria are satisfied, there is no guarantee that $v^{(j)}$ is close to the solution. The problem is that the simulator does not assure that KCL is satisfied. If progress on a particular iteration is slow, both (1) and (2) could be satisfied even though KCL is not. This condition is called false convergence. A good simulator will not use (2), but instead will check KCL directly.

The second criterion (2) allows Kirchhoff's current law to be violated slightly. In other words, the currents at each node do not quite sum to zero. This is equivalent to connecting small current sources to every node, randomly assigning current to these sources (with the proviso that the assigned current is smaller than that allowed by the convergence criterion), and solving this modified circuit exactly. This would be a problem with high-impedance nodes because even a small current injected into a high impedance contributes a significant voltage error. However, the first convergence criterion (1) limits the voltage error by insisting that the voltages converge. This criterion does not dictate the accuracy of the solution directly, because it compares the proposed solution against the value on the previous iteration, not the true solution. Thus, setting `reltol` to 0.001 does not imply the solution is accurate to 0.1%.

3.3 Transient Analysis

There are two important sources of error in a transient analysis besides those due to the models; errors due to nonideal convergence criteria and truncation error. The error due to the convergence criteria was discussed in the section on DC analysis. This results in charge not being completely conserved, which is an important point. Even simulators with charge-conserving models do not conserve charge completely, though they do a much better job of it than simulators that use non-charge-conserving models.

As mentioned before, circuits vary as how they accumulate error. Circuits that tend to be very sensitive to simulator errors include charge storage circuits such as switched-capacitor circuits and memories, chaotic circuits, such as oversampled analog/digital converters, and autonomous circuits such as oscillators. To determine whether a circuit is sensitive to simulator errors, ask yourself, if an error is made, will it dissipate or accumulate.

3.3.1 Truncation Error

The discretization or truncation error results from analyzing the circuit at a finite number of time-steps. Generally, more time-steps means less error, however they must be chosen carefully. It is very difficult to estimate the accuracy of transient analysis. It depends on the type of circuit being simulated, as well as on the number of time steps and their placement. Truncation error is controlled by `reltol`×`trtol`. The simulator chooses the time-step to control the truncation error made at each step. The amount of error is related to the step size. For first order methods such as backward Euler the error is proportional to the square of the step size (for small steps). For second order methods such as the trapezoidal rule and the second-order backward difference formula (Gear2), the error is proportional to the cube of the stepsize (again for small steps). How this error accumulates is determined by the circuit.

SPICE provides a method of time step control called "iteration-count time-step control," which chooses a time step based on the number of iterations required for the previous time-step. However, there is no direct relationship between iterations and truncation error. Thus, using this method essentially disables all control of the truncation error. For example, linear circuits always require only one iteration per time-step, but they still need their time-steps chosen to control truncation error. For this reason, *iteration-count time-step control (*`lvltim`*=1) should never be used* because the response computed by SPICE could be greatly in error.

The integration methods used in circuit simulation are taken from a general class of methods called multistep methods. These methods are all formulated to be exact for some low-order polynomial. For example, backward Euler is exact if the solution is a line. Trapezoidal rule and the second-order backward difference formula (Gear2) are exact for lines and quadratics. It is rare when solution waveforms follow linear or parabolic trajectories exactly, however over short time spans they can be approximated as such, and when using a quadratic the span can be longer than when using a line. An important practical point is that there is no truncation error when solution waveforms are constant valued. Thus, truncation errors affect time constants, such as settling times and periods of oscillation, but does not affect DC values or settled values unless they depend on time constants (for example, if an oscillator were to drive a frequency-to-voltage converter). The end result is that if you are very interested in finding equilibrium points accurately and the accuracy of time constants is not of paramount importance, you can get accurate results by tightening `reltol` and speed up the simulation by loosening `trtol`.

3.4 If Your Simulator Fails to Compute an Accurate Result

In general, if you would like your simulator to compute a more accurate solution, tighten `reltol`. Also, make sure `abstol` and `vntol` are reasonable. If your circuit is sensitive to simulator errors, it is a good idea to tighten `reltol` before you even start. In some situations tightening `reltol` may not help, or it may slow the simulator down more than necessary. So you might also consider the following suggestions.

3.4.1 DC Accuracy Problems

1. First assume that the simulator has computed the correct solution to the wrong circuit. Use your knowledge of the circuit to debug it using the computed DC solution and the operating point. Look for errors in the topology, the component parameters, the models, or the power supplies.

2. Assure that models you are using are appropriate and that the model parameters are consistent and correct.

3. Consider that your circuit might have more than one solution, which is fairly common, and the simulator found one you did not anticipate. Try using `nodeset` statements to

encourage the simulator to compute the solution you desire.

4. Tighten `reltol`. Also make sure `abstol` and `vntol` are reasonable.

5. Assure `gmin` is not affecting the solution. If possible, set `gmin` to zero.

3.4.2 Transient Accuracy Problems

Again, the best way to reduce the error in the solution in most situations is to tighten `reltol`. Artificially shrinking the time step is generally not a good idea because the Newton-Raphson convergence criteria are not affected and because not all time steps are shrunk, only the large ones. Generally you should only reduce the time step if you are convinced the results computed by the simulator are correct, but you want more time points for aesthetic reasons.

1. First verify that the circuit biased up properly. If not, there may be a problem with the topology, the models, or the power supplies.

2. Assure that models you are using are appropriate and that the model parameters are consistent and correct. Also check the operating point of each device.

3. Tighten `reltol`. Also make sure `abstol`, `vntol`, and `chgtol` are reasonable.

4. If you have a charge conservation problem, start by using a charge-conserving model. Then tighten `reltol` if more accuracy is needed. It is also possible for `abstol`, `chgtol`, and `gmin` to have an effect.

5. Assure `gmin` is not affecting the solution. If possible, set `gmin` to zero.

6. If the solution exhibits point-to-point ringing, switch the integration method to Gear's second-order backward-difference formula.

7. If a low-loss resonator seems to exhibit too much loss, switch the integration method to the trapezoidal rule.

8. If the simulator is not accurately following the turn-on transient of an oscillator, set the maximum timestep to at most one tenth the size of expected period of oscillation.

4 Circuits

In this section I discuss some of the important issues involved in simulating selected classes of circuits.

4.1 Oscillators

There are two issues that must be considered when simulating oscillators. First, it is necessary to manually start the oscillator. Second, oscillators are more sensitive to simulator error than most nonautonomous circuits, and so it is necessary to be more careful.

The DC analysis finds an equilibrium point for the circuit. By definition, the circuit will not drift away from an equilibrium point unless it is perturbed. If the equilibrium point is stable and the perturbation is small, the circuit will drift back to the equilibrium point afterwards. Oscillators have unstable equilibrium points. Thus, when an oscillator is perturbed from its equilibrium point it will break into oscillation rather return to the equilibrium point. However, a perturbation must be supplied to cause the oscillator to drift away from its equilibrium point and oscillate. In a physical circuit, thermal noise or the turn-on transients are sufficient to start the oscillator. In a circuit simulator, neither stimulus exists and so some explicit perturbation must be supplied by the user. While it is possible to use a independent source to start the oscillator, a generally easier, more reliable, and more controllable method is to set an initial condition on one of the resonator components.

Once an oscillator is perturbed, the oscillation will grow and approach its steady-state amplitude asymptotically. If the oscillator is started with a very small perturbation, then during the initial portion of the growth phase, the oscillation could be quite small. If the signals are not of sufficient size to force the timestep control algorithm to take smaller timesteps, then the simulator could portray the initial growth phase poorly. It is even possible that large timesteps could result in the effect of the perturbation being lost and having the oscillator fail to move away from its equilibrium point. In order to accurately predict the solution during its initial growth phase, it is necessary to force the simulator to use small timesteps using the maximum time-step parameter on the transient analysis. Typically, a reasonable value for maximum time-step would be

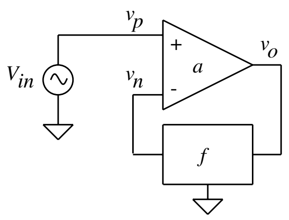

Figure 1: Series-shunt feedback circuit.

about one-tenth of the expected period of oscillation.

Oscillators generally have very high-Q resonators by design. High-Q resonators by their very nature can store energy for very long periods of time. Thus, errors created during a simulation will tend to be stored and accumulated in the resonator. Generally, the error that accumulates in the resonator results in an error in the period of oscillation. Thus, it is best when simulating oscillators to be conservative and tighten `reltol`, say to 0.0001. Lastly, you may want to consider using trapezoidal rule when simulating an oscillator because it does not exhibit artificial numerical damping like Gear2.

4.2 Operational Amplifiers

There are four quantities that are most useful when characterizing a feedback amplifier, closed-loop gain, open-loop gain, loop-gain, and feedback factor. I will now present a simple and direct approach to measuring these important quantities with a single AC analysis that does not require that the loop be broken. For the series-shunt feedback amplifier shown in figure 1 the feedback quantities are computed as follows:

Series-Shunt Configuration:

$A = v_o/v_p$	— closed-loop gain
$a = v_o/(v_p - v_n)$	— open-loop gain
$T = v_n/(v_p - v_n)$	— loop gain
$f = v_n/v_o$	— feedback factor

By substituting the appropriate current for the voltages given in the above equations, they can apply to the other three feedback configurations:

Series-Series Configuration:

$A = i_o/v_p$	— closed-loop gain
$a = i_o/(v_p - v_n)$	— open-loop gain
$T = v_n/(v_p - v_n)$	— loop gain
$f = v_n/i_o$	— feedback factor

Shunt-Series Configuration:

$A = i_o/i_p$	— closed-loop gain
$a = i_o/(i_p - i_n)$	— open-loop gain
$T = i_n/(i_p - i_n)$	— loop gain
$f = i_n/i_o$	— feedback factor

Shunt-Shunt Configuration:

$A = v_o/i_p$	— closed-loop gain
$a = v_o/(i_p - i_n)$	— open-loop gain
$T = i_n/(i_p - i_n)$	— loop gain
$f = i_n/v_o$	— feedback factor

The desirable features of this approach include:

1. No changes to the circuit are required.

2. All four measurements can be made from the results of a single AC analysis.

3. The loading on the various sections of the circuit does not change, as will occur if the loop is broken in any fashion.

4. The DC operating point remains unchanged.

This approach to measuring loop gain is superior to several other possible approaches. For example, the most obvious approach to measuring the loop gain is to open the loop. However, opening the loop changes the loading on the feedback circuit and often changes the operating point. The effect of either of these changes could invalidate the results.

A more refined approach is to open the loop in such a way that the DC operating point is not changed. Some simulators provide a resistor that takes different values in the DC and AC analyses. To measure loop gain, the resistor would be placed in series with the feedback loop and takes the value of 0 Ohms in the DC analysis (so the loop is closed when computing the DC operating point) and takes the value of infinity Ohms in the AC analysis (so the loop is open when measuring the loop gain). While this avoids a possible shift in the operating point, the loading still changes when the loop is opened during the AC analysis. This results in the computed value of loop gain being in error, particularly at high frequencies where the capacitive loading of the input on the output could be significant.