SEQUEL Users' Manual: Part 2

Mahesh B. Patil

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai-400076
e-mail: mbpatil@ee.iitb.ac.in

About Part 2 of the Manual

Simulating a circuit involves various steps: (i) creating the circuit schematic, (ii) assigning component values, (iii) choosing analysis type and assigning related parameters, (iv) running the simulator and viewing the results. SEQUEL provides a GUI for all of these steps. Video tutorials are available at the SEQUEL site to help a new user with the entire process in a step-by-step fashion.

The third task in the above list is achieved by defining "solve blocks." The purpose of Part 2 of the SEQUEL manual is to describe the syntax of the solve block statements. Before proceeding to the syntax rules, however, a few remarks are in order.

SEQUEL has been developed to simulate different kinds of circuits or systems: analog circuits, digital circuits, mixed-signal circuits, power electronic circuits including drives. If a simulator has to cater to a wide variety of circuits, it is difficult to set the algorithm/method parameters in a one-size-fits-all manner. For this reason, SEQUEL sets the method parameters to reasonable default values but allows the user to set them to suit his/her simulation need. Unfortunately, it calls for more work for the users since they need to know what each parameter means. The list of method parameters may seem intimidating at first, but it gets better after reading Part 1 of the SEQUEL manual. Several circuit files are provided with the SEQUEL distribution, and it is likely that the user will find an example similar to the circuit s/he wants to simulate. In such a case, the method parameters can be simply copied from that example, and chances are high that they will work without any changes.

There is a positive side to knowing a simulator in depth. Yes, it takes some effort, but it is a rewarding and enriching exercise. Apart from that, it would help the user in making better, more effective use of circuit simulators in general.

As Prof. V. Ramanarayanan of the Indian Institute of Science would put it, using a ready-made program is like eating canned food whereas writing your own program is like cooking your own meal, the way *you* would like it. Canned food has its advantages – it is quick and probably good enough for the average taste buds. Cooking yourself is more challenging – you need to start with chopping the ingredients and wait around when things get cooked or fried. But then you have complete control over the final product, and if all goes well, you have the satisfaction of doing a good job!

In the present context, understanding the functioning of a simulator and then using it is somewhere between eating canned food and preparing your own meal. With that brief introduction, let us get started.

In the following, we will group together solve block statements related to the same phenomenon or technique. That will make it easier to get an overall picture. Whenever applicable, we will cite equations or sections from Part 1 of the SEQUEL manual (as Sec. `xx` of Part-1, etc.) so that the reader can quickly make a connection between the solve block statements and the underlying method.

1 Newton-Raphson Method Parameters

The Newton-Raphson (NR) method, described in Chapter 3 of Part-1, is used in a variety of situations: DC, transient, SSW, and start-up (see Chapters 3, 6, 7, 8 in Part-1). The NR process is the same in all of these situations, and the NR method parameters are therefore common[1]. In the following, we describe these parameters.

* `itmax_newton`: (integer) maximum number of NR iterations (default: 500). Typically, the NR method converges in less than ten iterations, but the default value of `itmax_newton` has been made large to take care of special cases for which convergence is very slow (e.g., when there are exponential functions in the circuit/system being simulated, and the initial guess is poor.).

  In transient simulation, when the `back_euler_auto` or `trapezoidal_auto` option is used, `itmax_newton` should be set to a much smaller number, say, 5.

* `dmp`: (`yes`/`no`) decides whether damping (see Eq. 3.19 in Part-1) should be used (default: `no`)

* `dmp_k`: (real number) damping factor k where $0 < k < 1$ (see Eq. 3.19 in Part-1, default: 0.2). Not relevant when `dmp` is set to `no`.

* `dmp_newt_max`: (integer) number of NR iterations for which damping is applied (default: 50). Not relevant when `dmp` is set to `no`.

* `chk_rhs2`: (`yes`/`no`) decides whether the 2-norm (see Eq. 3.9 in Part-1) should be used to check for convergence. If there are electrical elements in the system, `chk_rhs2` is set to `no` by default; otherwise, it is set to `yes`.

* `chk_only_rhs2`: (`yes`/`no`) Setting this flag to `yes` is equivalent to setting `chk_rhs2` to `yes` and all other covergence flags to `no`.

* `norm_2`: (real number) tolerance value for the 2-norm (default: 10^{-10}). Not relevant when `chk_rhs2` is `no`.

* `write_rhs2`: (`yes`/`no`) decides whether the 2-norm should be written to the console (for each NR iteration). default: `no`.

* `chk_delx_volt`: (`yes`/`no`) decides whether ΔV, the node voltage difference between successive NR iterations, should be used to check for convergence (see Sec. 3.3 in Part-1). Default: `no`.

* `delxmax_volt`: (real number) tolerance value for ΔV (default (in Volts): 10^{-4}). Not relevant when `chk_delx_volt` is `no`.

* `write_delx_volt`: (`yes`/`no`) decides whether $\Delta V^{\max}$ should be written to the console (for each NR iteration). default: `no`.

* `chk_only_delx_volt`: (`yes`/`no`) Setting this flag to `yes` is equivalent to setting `chk_delx_volt` to `yes` and all other covergence flags to `no`.

[1]There are some exceptions such as NR parameters for the outer loop in SSW analysis (see Fig. 7.3 in Part-1) and NR parameters used during $g_{\min}$ stepping (see Sec. 3.5.2 in Part-1); these parameters are described separately.

* `chk_spice`: (`yes`/`no`) decides whether the SPICE convergence criteria (see Sec. 3.3 in Part-1) should be used to check for convergence. If there are electrical elements in the circuit/system, `chk_spice` is set to `yes` by default; otherwise, it is set to `no`.

* `chk_only_spice`: (`yes`/`no`) Setting this flag to `yes` is equivalent to setting `chk_spice` to `yes` and all other covergence flags to `no`.

* `norm_spice_rel`: (real number) k_{rel} in Eq. 3.10 of Part-1 (default: 10^{-3}).

* `norm_spice_nodev`: (real number) τ_{abs} for node voltages (see Eq. 3.10 of Part-1, default in Volts: 10^{-6}).

* `norm_spice_cur`: (real number) τ_{abs} for currents (default in Amps: 10^{-12}).

* `norm_spice_eaux`: (real number) τ_{abs} for EBE auxiliary variables (default: 10^{-4}).

* `norm_spice_locvar`: (real number) τ_{abs} for EBE local variables (default: 10^{-4}).

* `norm_spice_gvar`: (real number) τ_{abs} for GBE main variables (default: 10^{-4}).

* `norm_spice_gaux`: (real number) τ_{abs} for GBE auxiliary variables (default: 10^{-4}).

  In general, it is difficult to set `norm_spice_eaux`, `norm_spice_locvar`, `norm_spice_gvar`, and `norm_spice_gaux` in a meaningful manner because they correspond to variables of different kinds. For example, an auxiliary variable in an EBE may be a voltage or a current or a charge. They are made available to the user mainly for the sake of completeness.

* `write_spice`: (`yes`/`no`) decides whether information about SPICE convergence parameters should be written to the console (for each NR iteration). Default: `no`.

As we have seen in Chapter 3 of Part-1, convergence of the NR process depends on the initial guess. Convergence at the very first time point in transient simulation is more difficult because we may not have a good initial guess to start the NR process. For subsequent time points, the solution obtained at the previous time point generally serves as an excellent initial guess, and convergence is easier. For this reason, NR parameters for the first solution are made available separately, as given below.

* `itmax_newton_first`: (integer) maximum number of NR iterations for the first solution (default: 500).

* `dmp_first`: (`yes`/`no`) decides whether damping should be used for the first solution (default: `no`)

* `dmp_k_first`: (real number) damping factor k ($0 < k < 1$) for the first solution (default: 0.1). Not relevant when `dmp_first` is set to `no`.

* `dmp_newt_max_first`: (integer) number of NR iterations for which damping is applied for the first solution (default: 50). Not relevant when `dmp_first` is set to `no`.

2 Parameters related to $g_{\min}$ stepping

When the Newton-Raphson (NR) method fails to converge, SEQUEL uses $g_{\min}$ stepping (see Sec. 3.5.2 in Part-1). $g_{\min}$ stepping is allowed only for DC and transient simulation types. The following parameters are related to $g_{\min}$ stepping.

* `gmin_step`: (`yes`/`no`) decides whether $g_{\min}$ stepping should be used (when normal NR convergence fails). By default, this parameter is set to `yes` if there are highly nonlinear elements (semiconductor devices such as diodes and transistors) in the circuit; otherwise, it is set to `no`.

* `gmin_init`: (`yes`/`no`) decides whether $g_{\min}$ stepping should be used to obtain the initial solution (default: `yes`). Not relevant if `gmin_step` is `no`.

* `gmin_start`: The starting value of $g_{\min}$ in $\mho$ (default: 0.1)

* `gmin_end`: The final value of $g_{\min}$ in $\mho$ (default: 10^{-12})

* `gmin_npoints`: (integer) number of gmin points (default: 50).

  During $g_{\min}$ stepping, successive values of $g_{\min}$ are related by $g_{\min}^{(n+1)} = k' \times g_{\min}^{(n)}$. `gmin_start`, `gmin_end`, and `gmin_npoints` are used to compute the ratio k'.

* `gmin_itmax_newton`: (integer) maximum number of NR iterations allowed during $g_{\min}$ stepping (default: 500).

* `gmin_dmp`: (`yes`/`no`) decides whether damping (see Eq. 3.19 in Part-1) should be used during $g_{\min}$ stepping (default: `no`)

* `gmin_dmp_k`: (real number) damping factor k ($0 < k < 1$) to be used during $g_{\min}$ stepping (see Eq. 3.19 in Part-1). Not relevant when `gmin_dmp` is set to `no`.

* `gmin_dmp_newt_max`: (integer) number of NR iterations for which damping is applied during $g_{\min}$ stepping (default: 50). Not relevant when `gmin_dmp` is set to `no`.

3 Parameters related to SSW analysis

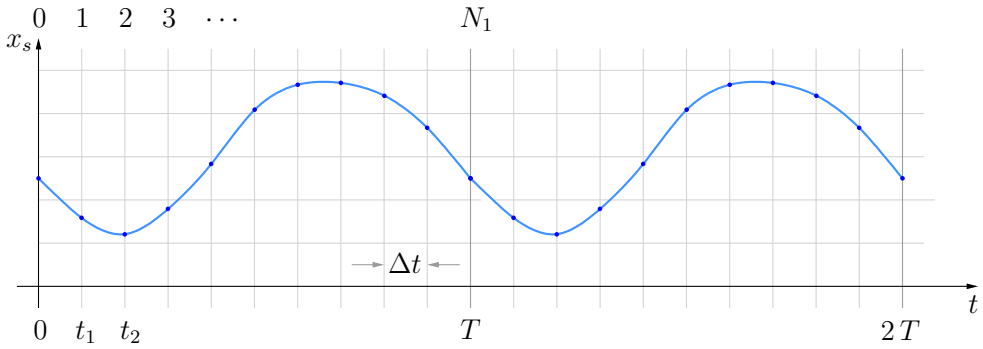

Figure 1: Illustration of parameters related to SSW analysis. A schematic plot of $x_s(t)$ is shown where x_s is a state variable.

SSW analysis is meant for computing the periodic steady-state solution (see Chapter 7 of Part-1). There are two sets of parameters related to SSW analysis: (a) waveform parameters, (b) NR parameters.

Waveform parameters:

* `ssw_period`: (real number) waveform period (T in the figure).
* `ssw_frequency`: (real number) waveform frequency ($1/T$).

  Either `ssw_period` or `ssw_frequency` must be specified.
* `ssw_ndiv`: (integer) number of divisions in one period (N_1 in the figure, default: 100). If `delt_const` (Δt in the figure) is specified, `ssw_ndiv` is ignored.
* `ssw_period_mult`: (integer) number of SSW periods to be simulated (default: 1). Only one period is really required to be simulated. However, the user may want to view a graph showing a few periods, and therefore this option is made available.

NR parameters: As shown in Fig. 7.3 in Part-1, the SSW state variable computation involves a Newton-Raphson (NR) loop (the *outer* NR loop in the figure). Parameters related to this loop are specific to SSW analysis and are described below[2].

* `ssw_itmax_newton`: (integer) maximum number of outer NR iterations allowed (default: 20).
* `ssw_dmp`: (`yes`/`no`) decides whether damping (see Eq. 3.19 in Part-1) should be used for the outer NR loop (default: `no`)
* `ssw_dmp_k`: (real number) damping factor k ($0 < k < 1$) for the outer NR loop (see Eq. 3.19 in Part-1, default: 0.8). Not relevant when `ssw_dmp` is set to `no`.

[2]If the circuit is nonlinear, there is *another* NR loop – the *inner* NR loop. Parameters for that loop (e.g., `itmax_newton`, `dmp`, etc.) are identical to those described elsewhere and are not repeated here.

* `ssw_dmp_newt_max`: (integer) number of NR iterations for which damping is to be applied in the outer NR loop (default: 10). Not relevant when `ssw_dmp` is set to `no`.

* `ssw_chk_rhs2`: (`yes`/`no`) specifies whether the 2-norm should be used to check for convergence of the outer NR iterations (default: `no` if there are electrical elements in the system being simulated; else, `yes`).

* `ssw_norm`: (real number) specifies the tolerance to check convergence of the outer NR iterations. It applies if `ssw_chk_rhs2` is `yes`, and in that case, the 2-norm, $\left[\frac{1}{N'}\sum_{i=1}^{N'} f_i\right]^{1/2}$, is compared with `ssw_norm` to test for convergence, where N' is the number of state variables.

* `ssw_chk_spice`: (`yes`/`no`) specifies whether the SPICE convergence criteria should be used to check for convergence of the outer NR iterations (default: `yes` if there are electrical elements in the system being simulated; else, `no`).

4 Statements related to frequency specification

An AC (frequency-domain) solution requires the frequency (frequencies) to be specified (see Chapter 9 of Part-1). SEQUEL allows the following options in that context.

* `set_frequency R`
  (R: real number)
  corresponds to the GUI statement `set frequency`. This statement is used to instruct the simulator to perform AC simulation at a single frequency `R`.

* `vary_freq from R1 to R2 type=linear/log n_points=I`
  (`R1,R2`: real numbers, `I`: integer)
  corresponds to the GUI statement `vary frequency`. This statement is used to instruct the simulator to perform AC simulation for a range of frequencies from `R1` to `R2`. The number of frequency points is specified by `I`.

  If the type is specified as `linear`, the frequency values are assumed to be distributed linearly between `R1` and `R2`.

  If the type is specified as `log`, the frequency values are assumed to be distributed logarithmically between `R1` and `R2`.

  In typical applications, `R1` and `R2` are orders of magnitude apart, and the `log` option is more appropriate.

* `vary_freq type=table R1 R2 R3 ...`
  (`R1,R2,R3,...`: real numbers)
  corresponds to the GUI statement `vary frequency`. This statement is used to instruct the simulator to perform AC simulation at frequencies given by `R1`, `R2`, `R3`,....

In an AC simulation solve block, either `set_freq` or `vary_freq` must be included.

5 Statements related to setting parameters

The SEQUEL GUI allows element parameters to be set with the "property editor." In addition, it is sometimes desirable to set a parameter of a given element inside a solve block, set it to another value in another solve block, and so on. The following statements may be used for that purpose.

* `set_parm S1_of_S2=I/R`
  (`S1,S2`: strings, `I`: integer, `R`: real number)
  corresponds to the GUI statement `set parameter`. This statement is used to set an integer or real parameter of an element. The parameter name is given by `S1`, and the element name by `S2`. The parameter value is specified by `I` or `R`.

* `set_parm S1_of_glbl=I/R`
  (`S1`: string, `I`: integer, `R`: real number)
  corresponds to the GUI statement `set global parameter`. This statement is used to set an integer or real global parameter `S1`. The parameter value is specified by `I` or `R`.

* `set_stparm S1_of_S2=R`
  (`S1,S2`: strings, `R`: real number)
  corresponds to the GUI statement `set startup parameter`. This statement is used to set a start-up parameter of an element (such as voltage across a capacitor or current through an inductor, see Chapter 8 of Part-1). The parameter name is given by `S1`, and the element name by `S2`. The parameter value is specified by `R`.

6 Statements related to varying parameters

The SEQUEL GUI allows element parameters to be set with the "property editor." In addition, it is sometimes desirable to vary a parameter of a given element from a starting value to an ending value inside a solve block. For example, in generating the *I-V* curve of a device, we would like to vary the voltage across the device and record the current. The following statements may be used to vary parameters.

* `vary_parm S1_of_S2 from R1 to R2 type=linear n_points=I1`
  (`S1,S2`: strings, `I1`: integer, `R1,R2`: real numbers)
  corresponds to the GUI statement `vary parameter (NPoints)`. This statement is used to vary a real parameter of an element from `R1` to `R2` in a linear fashion. The parameter name is given by `S1`, and the element name by `S2`. The number of parameter values is specified by `I1`.

  For example, `vary_parm vdc_of_Vs from 0 to 5 type=linear n_points=51`
  can be used to vary the parameter `vdc` of the element `Vs` from 0 to 5. Since `n_points` is specified as 51, the interval $(5 - 0)$ is divided into $(51 - 1)$ intervals (i.e., each interval equal to 0.1), and the parameter is varied as 0, 0.1, 0.2, $\cdots$, 4.9, 5.0.

* `vary_parm S1_of_S2 from R1 to R2 type=log n_points=I1`
  (`S1,S2`: strings, `I1`: integer, `R1,R2`: real numbers)
  corresponds to the GUI statement `vary parameter (NPoints)`. This statement is used to vary a real parameter of an element from `R1` to `R2` in a logarithmic fashion. The parameter name is given by `S1`, and the element name by `S2`. The number of parameter values is specified by `I1`.

* `vary_parm S1_of_S2 type=table R1 R2 R3 ...`
  (`S1,S2`: strings, `R1,R2,R3,...`: real numbers)
  corresponds to the GUI statement `vary parameter (NPoints)`. This statement is used to vary a real parameter of an element. The parameter values to be assigned are given by `R1`, `R2`, `R3`,.... The parameter name is given by `S1`, and the element name by `S2`. Note that `n_points` is not relevant in this case.

* `vary_parm S1_of_S2 from R1 to R2 type=linear div=R3`
  (`S1,S2`: strings, `R1,R2,R3`: real numbers)
  corresponds to the GUI statement `vary parameter (Div)`. This statement is used to vary a real parameter of an element from `R1` to `R2` in a linear fashion. The parameter name is given by `S1`, and the element name by `S2`. The interval between successive parameter values is specified by `R3`.

  For example, `vary_parm vdc_of_Vs from 0 to 5 type=linear div=0.2`
  can be used to vary the parameter `vdc` of the element `Vs` from 0 to 5. Since `div` is specified as 0.2, the parameter is varied as 0, 0.2, 0.4, $\cdots$, 4.8, 5.0.

Note: The `vary parameter` statement is not allowed in transient and SSW simulation.

7 Statements related to varying global parameters

The SEQUEL GUI allows global parameter values to be assigned with the "property editor." In addition, it is sometimes desirable to vary a global parameter from a starting value to an ending value inside a solve block. The following statements may be used for that purpose.

* `vary_parm S1_of_glbl from R1 to R2 type=linear n_points=I1`
  (`S1`: string, `I1`: integer, `R1,R2`: real numbers)
  corresponds to the GUI statement `vary parameter global`. This statement is used to vary a global real parameter from `R1` to `R2` in a linear fashion. The parameter name is given by `S1`. The number of parameter values is specified by `I1`.

* `vary_parm S1_of_glbl from R1 to R2 type=log n_points=I1`
  (`S1`: string, `I1`: integer, `R1,R2`: real numbers)
  corresponds to the GUI statement `vary parameter global`. This statement is used to vary a global real parameter from `R1` to `R2` in a logarithmic fashion. The parameter name is given by `S1`. The number of parameter values is specified by `I1`.

* `vary_parm S1_of_glbl type=table R1 R2 R3 ...`
  (`S1`: string, `R1,R2,R3,...`: real numbers)
  corresponds to the GUI statement `vary parameter global`. This statement is used to vary a global real parameter. The parameter values to be assigned are given by `R1`, `R2`, `R3`,.... The parameter name is given by `S1`. Note that `n_points` is not relevant in this case.

Note: The `vary parameter global` statement is not allowed in transient and SSW simulation.

8 Statements related to initial solution

SEQUEL allows the following options for generating the initial solution, which serves as the starting point for the Newton-Raphson process (in a nonlinear problem).

* `initial_sol initialize`
  When this option is selected, all variables are initialised to zero to generate the initial solution. This is the default option.

* `initial_sol previous`
  When this option is selected, the solution computed in the previous solve block is used as the initial solution.

* `initial_sol file filename=S1`
  (`S1`: string)
  When this option is selected, the initial solution is read from a file. The string `filename` specifies the name of the file, e.g., `x1.dat` (without quotation marks).

9 Output block

The output block is used to convey to the simulator several details such as which output files to generate, which variables to include in each file, etc. When the program executes successfully, the user-specified output files get created. To view the information contained in the output files (as a plot or a table), the SEQUEL GUI or any other plotting package can be used.

General attributes

* `FileName`: name of the output file (default: `output.dat`)
* `Output Variables`: output variables to be stored in the file.
* `LimitLines`: (integer) maximum number of lines to be stored. This is a "safety feature" to ensure that the user does not unknowingly generate very large files. If the number of lines generated by the program exceeds `LimitLines`, SEQUEL will produce an error message. If there is a genuine requirement of a large amount of data points, the user should increase `LimitLines` suitably. Default: 100,000.
* `Append`: (`yes`/`no`) decides whether the output data for the present output file should be appended to previously existing data (default: `no`).

  As an example, suppose that we have split a transient simulation from t_1 to t_3 to two intervals (i.e., two solve blocks): (a) t_1 to t_2 and (b) t_2 to t_3. We want a variable to be recorded for both these intervals in the *same* output file. In this case, we would make the output file names identical in the two solve blocks and set `Append` to `yes` in the second solve block.

Attributes related to transient simulation

* `FixedInterval`: (real number) if specified, the output variables are recorded at uniform intervals.

  For example, if `50u` is specified, the output variables are recorded every 50 μsec.
* `OutTStart`: (real number) if specified, the output of the output variables are recorded only for $t >$ `OutTStart`.
* `OutTEnd`: (real number) if specified, the output of the output variables are recorded only for $t <$ `OutTEnd`.

  if `OutTStart` and `OutTEnd` are not specified, the output variables are recorded for the entire simulation interval (from `t_start` to `t_end`).
* `Fourier`: (`yes`/`no`) decides whether Fourier components of the output variables should be computed and stored (default: `no`). If `Fourier` is specified as `yes`, the waveforms are assumed to be periodic with the period T computed as the difference `OutTEnd` − `OutTStart`.

  When `Fourier` is specified as `yes`, the total harmonic distortion (THD) is also made available (in the `Solver Output` tab) for the variables listed in the output block. The following definition of THD is used.

$$\text{THD} = \frac{\sqrt{X_2^2 + X_3^2 + \cdots}}{X_1}, \tag{1}$$

where X_1, X_2, X_3 (etc.) are the f, $2f$, $3f$ components, respectively, of the concerned variable.

* `NFourier`: (integer) number of Fourier components to be computed.

Attributes related to AC simulation

* `MinPhase`: (real number) If `MinPhase` is specified, the phase data is written to the output file such that the phase angle is always larger than `MinPhase` (by adding suitable multiples of 360°).

* `MaxPhase`: (real number) If `MaxPhase` is specified, the phase data is written to the output file such that the phase angle is always smaller than `MaxPhase` (by adding suitable multiples of 360°).

  Note that either `MinPhase` or `MaxPhase` can be specified, but not both.

* `FreqHz`: (`yes`/`no`). If `yes`, the frequency values are written to the output file in Hz; else, in rad/s (default: `yes`).

10 `offs_dgtl`

Digital variables take on only two values: `0` and `1`. If several digital variables are plotted versus time in the same plot, it is difficult to view them (see Fig. 2, top plot). The parameter `offs_dgtl` (real number) is used to introduce an offset between successive variables (see Fig. 2, bottom plot).

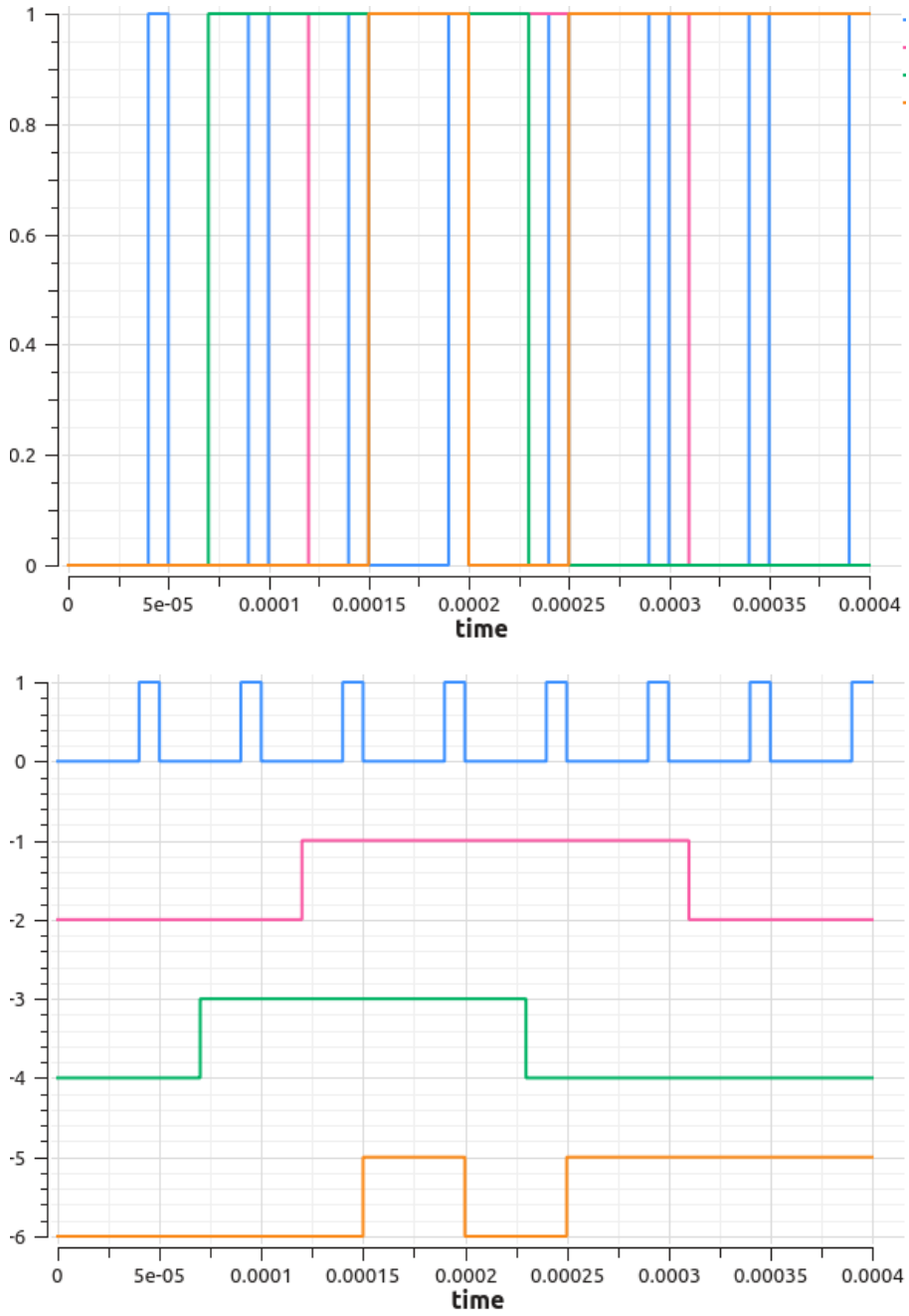

Figure 2: Example of digital output waveforms: `offs_dgtl` = 0 (top plot), `offs_dgtl` = −2 (bottom plot).

11 Parameters related to transient simulation

In the following, we describe parameters related to transient simulation (see Chapter 6 in Part-1). Some of the parameters would also apply to SSW computation since SSW involves transient simulation as well (see Fig. 7.3 in Part-1).

* `back_euler`: (`yes`/`no`) for Backward Euler method with constant time step (given by `delt_const`). In this case, a few smaller time steps may be taken to account for corners in input waveforms, for example.

* `back_euler_auto`: (`yes`/`no`) for Backward Euler method with NR-based adaptive time steps (see Sec. 6.6 in Part-1)

* `trapezoidal`: (`yes`/`no`) for trapezoidal method with constant time step (given by `delt_const`). In this case, a few smaller time steps may be taken to account for corners in input waveforms, for example.

* `trapezoidal_auto`: (`yes`/`no`) for trapezoidal method with NR-based adaptive time steps (see Sec. 6.6 in Part-1)

* `gear2`: (`yes`/`no`) for second-order Gear (BDF) method with constant time step (not available for SSW)

* `trbdf2`: (`yes`/`no`) for TR-BDF2 method (not available for SSW)

* `constant_step`: (`yes`/`no`) for forcing constant time steps. In this case, the step size is always equal to `delt_const`. If `constant_step` is selected, `back_euler` or `trapezoidal` must also be selected. (not available for SSW)

* `t_start`: (real number) starting time for transient simulation

* `t_end`: (real number) ending time for transient simulation

* `itmax_trns`: (integer) maximum number of time points (default: 100,000). This is a "safety feature." If the user by mistake creates conditions which calls for a large number of time points, SEQUEL will produce an error message. If there is a genuine requirement, the user should increase `itmax_trns` suitably.

* `delt_const`: (real number) serves as the constant time step for BE and TRZ methods, and as the first time step for methods with adaptive time steps.

* `delt_min`: (real number) smallest time step allowed (default: $0.0002 \times$ `delt_const`)

* `delt_max`: (real number) largest time step allowed (default: $10 \times$ `delt_const`)

* `fctr_stepred`: (real number) factor for reducing time step in adaptive time step methods (k_{down} in Sec. 6.6 in Part-1, default: 0.6)

* `fctr_stepinc`: (real number) factor for increasing time step in adaptive time step methods (k_{up} in Sec. 6.6 in Part-1, default: 1.5)

* `itmax_stepred`: (integer) maximum number of successive reductions in the time step at a given time point (default: 20)

* `trbdf2_tolr`: (real number) tolerance for TR-BDF2 method (default: 10^{-5})

* `itmax_trbdf2`: (integer) maximum number of successive reductions in the time step at a given time point when TR-BDF2 method is used (default: 20)

12 Parameters related to explicit methods

For transient simulation of a circuit with explicit compound elements (XCEs), SEQUEL employs explicit methods[3] (see Chapter 4 of Part-1). The following parameters apply in that case.

* `forward_euler`: (`yes`/`no`) for Forward Euler method with constant time step.
* `RK4`: (`yes`/`no`) for Runge-Kutta order-4 method with constant time step.
* `modified_euler`: (`yes`/`no`) for Modified (Improved) Euler method (see [1], for example) with constant time step.
* `Heun`: (`yes`/`no`) for Heun method (see [1], for example) with constant time step.

  When a constant step method (Forward Euler, RK4, Modified Euler, Heun) is used, a few smaller time steps may be taken to account for corners in input waveforms, for example.
* `RKF45`: (`yes`/`no`) for Runge-Kutta-Fehlberg 4/5 method (auto time step)
* `BS23`: (`yes`/`no`) for Bogacki-Shampine 2/3 method [2] (auto time step)
* `delt_const_x`: (real number) serves as the constant time step for Forward Euler, RK4, Modified Euler, and Heun methods, and as the first time step for methods with auto time steps.
* `delt_min_x`: (real number) smallest time step allowed (default: $0.0002 \times$ `delt_const_x`)
* `delt_max_x`: (real number) largest time step allowed (default: $10 \times$ `delt_const_x`)

Next, we list parameters related to the RKF45 and BS23 methods. These methods use auto time stepping. At each time point, a multiplier k is computed from the tolerance and an estimate of the local truncation error (see Sec. 4.4 of Part-1) to obtain the next time step as

$$\Delta t^{\text{new}} = k \times \Delta t^{\text{old}}. \tag{2}$$

The parameters are

* `rkf45_tolr`: (real number) tolerance value for the RKF45 method (default: 10^{-8})
* `rkf45_fctr_min`: (real number) lower limit on multiplier k (Eq. 2) in the RKF45 method. (default: 0.8)
* `rkf45_fctr_max`: (real number) upper limit on multiplier k (Eq. 2) in the RKF45 method. (default: 1.1)
* `bs23_tolr`: (real number) tolerance value for the BS23 method (default: 10^{-8})
* `bs23_fctr_min`: (real number) lower limit on multiplier k (Eq. 2) in the BS23 method. (default: 0.8)
* `bs23_fctr_max`: (real number) upper limit on multiplier k (Eq. 2) in the BS23 method. (default: 1.1)

[3] With explicit elements, only transient simulation is allowed; DC, start-up, AC, SSW are not allowed.

As we have seen in Sec. 4.6 of Part-1, algebraic loops create a difficulty for explicit methods. If the user's system has algebraic loops, the algebraic equations must be solved separately. As an example, consider the system shown in Fig. 3 which has two algebraic loops (the loops involving multipliers k_1 and k_2).

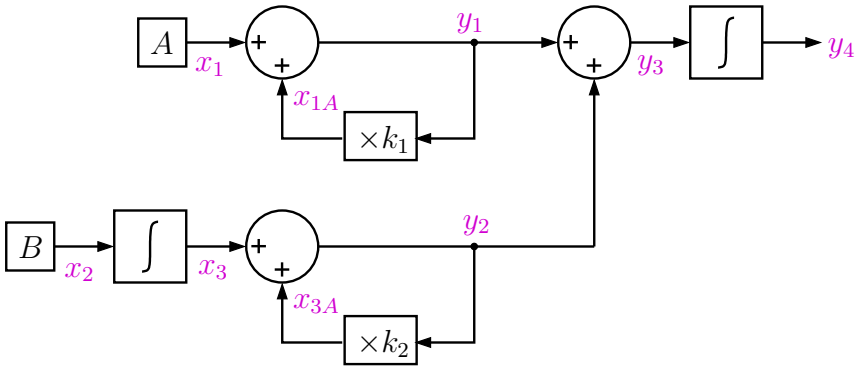

Figure 3: A system with algebraic loops. A and B are constants.

In such a case, SEQUEL would first update the variables associated with a time derivative, in this case, x_3 and y_4. For example, with the Forward Euler method, we have

$$y_4^{n+1} = y_4^n + hy_3^n, \tag{3}$$

$$x_3^{n+1} = x_3^n + hx_2^n, \tag{4}$$

where $h = t_{n+1} - t_n$ is the time step.

Having obtained y_4^{n+1} and x_3^{n+1}, the other variables are updated by solving the algebraic equations governing those variables. In the above example, all elements are linear, so it is a simple matter of solving a linear system of equations. If there are nonlinear elements in the system, the Newton-Raphson (NR) method (see Chapter 3 of Part-1) is used to solve the resulting equations.

Apart from algebraic loops, there is another situation in which a linear or nonlinear system of equations needs to be solved, and that is the presence of electrical-type elements which are implemented as XCEs. In that case, SEQUEL internally adds the required KCL and KVL equations.

The overall set of equations is divided into two sub-sets: (a) ODEs, (b) algebraic equations. An algebraic equation may arise from an algebraic loop, it may be a KCL/KVL equation, or it may be an element behaviour equation. The two sub-sets are treated separately – the ODEs with an explicit method (specified by the user) and the algebraic equations with a linear or nonlinear solver.

The following parameters are relevant in the above context.

* `x_eval_serial`: (`yes`/`no`) is used to indicate whether the elements should be evaluated in a serial fashion (see Sec. 4.6 in Part-1). It should be set to `no` when there are algebraic loops or electrical-type XCEs in the system.

  Default: If there are electrical-type XCEs, `x_eval_serial` is set to `no` by default; else, it is set to `yes`.

* `x_itmax_newton`: (integer) maximum number of NR iterations (default: 500).

* `x_dmp`: (`yes`/`no`) decides whether damping (see Eq. 3.19 in Part-1) should be used (default: `no`)

* `x_dmp_k`: (real number) damping factor k where $0 < k < 1$ (see Eq. 3.19 in Part-1, default: 0.2). Not relevant when `x_dmp` is set to `no`.

* `x_dmp_newt_max`: (integer) number of NR iterations for which damping is applied (default: 50). Not relevant when `x_dmp` is set to `no`.

* `x_chk_rhs2`: (`yes`/`no`) decides whether 2-norm (see Eq. 3.9 in Part-1) should be used to check for convergence of NR iterations. (default: `yes`)

* `x_norm_2`: (real number) tolerance value for the 2-norm (default: 10^{-10}). Not relevant when `x_chk_rhs2` is `no`.

* `x_write_rhs2`: (`yes`/`no`) decides whether the 2-norm should be written to the console (for each NR iteration). Default: `no`.

* `x_chk_spice`: (`yes`/`no`) decides whether the SPICE convergence criteria (see Sec. 3.3 in Part-1) should be used to check for convergence of NR iterations. (default: `no`)

* `x_norm_spice_rel`: (real number) k_{rel} in Eq. 3.10 of Part-1 (default: 10^{-3}).

* `x_norm_spice_nodev`: (real number) τ_{abs} for node voltages (see Eq. 3.10 of Part-1, default in Volts: 10^{-6}).

* `x_norm_spice_cur`: (real number) τ_{abs} for currents (default in Amps: 10^{-12}).

* `x_norm_spice_xaux`: (real number) τ_{abs} for XBE auxiliary variables (default: 10^{-4}).

  In general, it is difficult to set `x_norm_spice_xaux` in a meaningful manner because it corresponds to variables of different kinds. For example, an auxiliary variable in an XBE may be a speed or force or current. It is made available to the user mainly for the sake of completeness.

* `x_write_spice`: (`yes`/`no`) decides whether information about SPICE convergence parameters should be written to the console (for each NR iteration). Default: `no`.

As we have seen in Chapter 3 of Part-1, convergence of the NR process depends on the initial guess. Convergence at the very first time point in transient simulation is more difficult because we may not have a good initial guess to start the NR process. For subsequent time points, the solution obtained at the previous time point generally serves as an excellent initial guess, and convergence is easier. For this reason, NR parameters for the first solution are made available separately, as given below.

* `x_itmax_newton_first`: (integer) maximum number of NR iterations for the first solution (default: 500).

* `x_dmp_first`: (`yes`/`no`) decides whether damping should be used for the first solution (default: `no`)

* `x_dmp_k_first`: (real number) damping factor k ($0 < k < 1$) for the first solution (default: 0.1). Not relevant when `x_dmp_first` is set to `no`.

* `x_dmp_newt_max_first`: (integer) number of NR iterations for which damping is applied for the first solution (default: 50). Not relevant when `x_dmp_first` is set to `no`.

References

1. M. B. Patil, V. Ramanarayanan, and V. T. Ranganathan, *Simulation of Power Electronic Circuits*, Narosa, New Delhi, 2009.

2. `https://en.wikipedia.org/wiki/Bogacki%E2%80%93Shampine_method`