Appendix:

Table from the original SIMPSON paper (M. Bak. J. T. Rasmussen, and N. C. Nielsen, *J. Magn. Reson.* **147** (2000) 296) introducing the basic elements of a SIMPSON input file.

TABLE 2
Elements of and Scripting Commands for the SIMPSON Input File[a]

Elements of the SIMPSON input file

`spinsys {. . .}`	Spin system and interactions.
`par {. . .}`	Global experiment parameters.
`proc pulseq {} {. . .}`	Pulse sequence.
`proc main {} {. . .}`	Processing control.

Declarations for the `spinsys` section

`channels` N_1 N_2 ... N_n
`nuclei` N_1 N_2 ... N_n
`shift` i $\delta^i_{iso}|\omega^i_0/2\pi|^b$ $\delta^i_{aniso}|\omega^i_0/2\pi|^b$ η^{CS} α_{PC} β_{PC} γ_{PC}
`dipole` i j $b_{ij}/2\pi$ α_{PC} β_{PC} γ_{PC}
`jcoupling` i j J^{ij}_{iso} J^{ij}_{aniso} η^J α_{PC} β_{PC} γ_{PC}
`quadrupole` i $order^c$ $C_Q/2\pi$ η^Q α_{PC} β_{PC} γ_{PC}

Parameters/commands for the `par` section

`spin_rate`	Sample spinning frequency, $\omega_r/2\pi$
`np`	Number of sampling points.
`ni`	Number of sampling points in the indirect dimension of 2D experiments.
`sw`	Spectral width.
`sw1`	Spectral width for the indirect dimension.
`crystal_file`	Name of the powder averaging file containing the number N of orientations and α^k_{CR}, β^k_{CR}, ω_k values on successive lines.
`gamma_angles`	Number M of γ_{CR} angles. Set to 1 if `spin_rate` is zero. Defines the number of sampling points per rotor period when `method` equals `gcompute`.
`gamma_zero`	Constant value added to all γ_{CR} values in the powder averaging. Specifies the γ_{CR} angle if only a single crystallite is used.
`rotor_angle`	Angle between the rotor axis and the B_0 direction. If `spin_rate` is zero the default angle is zero, otherwise the magic angle.
`method`	Chooses among `direct`, `gammarep`, and `gcompute` methods for the simulation.
`start_operator`	$\rho(0)$ defined as an expression using operators $I_i\alpha$ where α is x, y, z, p (+) or m (−), and i is the nucleus number or n to denote the sum over all nuclei.[d]
`detect_operator`	Detection operator Q_{det}.[d]
`pulse_sequence`	Sets another name than `pulseq` for the pulse sequence.
`proton_frequency`	Absolute ^{1}H Larmor frequency $|\omega^H_0/2\pi|$ in hertz. Used for ppm to hertz convertion and for the second-order quadrupolar coupling.
`verbose`	A row of flags that sets the level of information printed when running the simulation.
`variable` *name*	Sets a user specific variable with a value that can be retrieved throughout the input file by declaring: `$par(`*name*`)`.

Commands for the `pulseq` section.

`pulse` δt $|\omega^1_{RF}/2\pi|$ ϕ_1 $|\omega^2_{RF}/2\pi|$ ϕ_2 ...
Extends the current propagator to include a pulse of duration δt, RF-field amplitude of $\omega^i_{RF}/2\pi$, and phase ϕ_i on the channels numbered successively. Alternatively the phase can be specified as x, y, $-x$, or $-y$ corresponding to phases of 0, 90, 180, or 270 degrees, respectively.

`pulseid` δt $|\omega^1_{RF}/2\pi|$ ϕ_1 $|\omega^2_{RF}/2\pi|$ ϕ_2 ...
Same as `pulse` but performs an ideal (i.e., infinitely strong and infinitely short) pulse. δt and $\omega^i_{RF}/2\pi$ have no physical meaning other than to specify the flip-angle of the pulse. The internal time remains unchanged.

`delay` δt
Extends the current propagator to include a free precession period of duration δt. If the Hamiltonian is diagonal (i.e., no homonuclear spin–spin couplings) the delay is calculated by analytical integration of the Hamiltonian.

`offset` $\omega^1_{off}/2\pi$ $\omega^2_{off}/2\pi$ ...
Invokes an offset of $\omega^i_{off}/2\pi$ Hz to the channels numbered successively. The offset for a channel is defined by the Hamiltonian $\omega^i_{off}\sum_j I_{jz}$, where j is summed over the nuclei in the spin-system affected by pulses on the channel. The offset applies until reset using offset with zero-value arguments or the pulse sequence is called again.

`acq` [*n prop*] [ϕ]
Propagates $\rho(t)$ using the current propagator, collects a data point corresponding to Q_{det}, and resets the propagator to unity. The optional arguments n and *prop* specifies the number of data points n to collect while evolving with propagator number *prop*. The optional argument ϕ specifies the receiver phase (syntax as for `pulse`).

`maxdt` Δt
Maximum time step (Δt in Eq. [4]) over which the Hamiltonian may be considered time independent. The computation time/accuracy of the simulation is significantly affected by the choice of value for this parameter. Defaults to 1 μs in case of sample spinning and infinity in the static case.

TABLE 2—*Continued*

`store` *n*
Stores the current propagator in memory slot number *n*. The current propagator is not reset.

`reset` [δt]
Resets $\rho(t)$ to the initial operator $\rho(0)$ and the current propagator to the unity operator. Resets the current time and adds δt μs if specified.

`prop` *n* [*times*]
Propagates $\rho(t)$ with the propagator saved in memory slot number *n*. Repeated multiple times if *times* is specified, skipped if *times* is zero, or propagated once if *times* is omitted.

`filter` *n*
Propagates $\rho(t)$ with the current propagator, after which the propagator is reset and elements in the density matrix are set to zero if the corresponding element in matrix number *n* (defined using `matrix set`) is zero.

`select` *n* ...
Renders the next pulse (`pulse` or `pulseid`) selective toward the spins which numbers are given as argument.

`turnoff` *int* ...
Disables the effect of the specified interactions until the end of the pulse sequence or until they are reactivated by `turnon`. Interactions are named *int_n* or *int_n_m* using names and numbers given in the `spinsys` section, or `all` if all interactions should be disabled.

`turnon` *int* ...
Enables the specified interactions or all interactions if `all` is specified.

int `getinteractions`
Returns a list of lists each containing an interaction name (see `turnoff`) and 1 or 0 depending on whether it is enabled or disabled.

`putmatrix` *matrix* ?*format*?
Prints out a matrix returned by `matrix get` optionally in a format different from the standard format "`%9.3g`".

`matrix set` *to from*
Sets a matrix *to* to the contents of a matrix created using the argument *from* described below. *to* can either be an index in the internal array of matrices, the start (`start`) or the detect (`detect`) operator.

matrix `matrix get` *from*
Returns a matrix (printed with `putmatrix`) based on the argument *from* which can be either the same as *to* described above or the Hamiltonian (`hamiltonian`), current propagator (`propagator`), current density operator (`density`), an operator expression (`operator` *expr*) or for the purpose of filtering (undesired elements set to zero) the specific total coherence orders (`totalcoherence {. . .}` with the list containing coherences), coherence orders (`coherence {{. . .} . . .}` with each sublist containing coherence orders for each nuclei), the full matrix (`list {row row . . .}` where each row is a list of elements being either *re* or {*re im*}), specific matrix elements (`elements {{*i j*} ...}`), or all elements excluding specific matrix elements (`notelements {{*i j*} ...}`).

Commands for the `main` section

d `fsimpson` [{{*int_n_nam v*} ...}]
Starts a simulation and returns a data set `d`, optionally overriding specific values of the interactions given in the `spinsys` section. A value is named *int_n_nam* or *int_n_m_nam*, where *int* is the interaction name from `spinsys`, *n* and/or *m* the numbers of the involved nuclei, and *nam* is `iso`, `aniso`, `eta`, `alpha`, `beta`, or `gamma`.

`fsave` *d file* [*-format* `-binary -double`]
Saves the (possibly 2D) data set *d* to a file *file* using the SIMPSON data format in text, binary (`-binary`) single or double (`-double`) precision format, or optionally in another format (*-format*) being (1) `-xreim` with rows of frequency/time, real and imaginary part of data, (2) `-xyreim` (2D data) with rows of frequency/time (indirect dimension), frequency/time (direct dimension), real and imaginary part of data, with an empty line separating succeeding fids, or (3) `-gnu2d -binary` in the binary 2D Gnuplot format (*56*).

`fft` *d* [`-inv`]

`fft` *d rp lp rp1 lp1* [`-phsens`]
The first form performs a direct or inverse fast Fourier transformation of the data set *d*, while the second form performs a 2D transformation using constant (*rp*) and linear (*lp*) phase correction in the direct and indirect (*rp1* and *lp1*) dimensions. The optional argument `-phsens` assumes phase-sensitive 2D data with succeeding pairs of fids corresponding to equal t_1 and 90° different phase.

`fzerofill` *d npz* [*niz*]
Zerofills the data set *d* up to a total of *npz* points, optionally zerofills the 2D data set up to a total of *niz* points in the indirect dimension.

`fphase` *d* [`-rp` *v* `-lp` *v* `-scale` *v* `-offset` *v*]
Performs one or more of first- and second-order phasing vertical scaling, and offset on the data set *d*.

`faddlb` *d lb r* [*lb1 r1* [`-phsens`]]
Apodizes the data set *d* with a Gaussian/Lorentzian (ratio *r*) weighting function causing an extra linebroadening of *lb* Hz, or optionally the 2D data set with *lb1* and *r1* specifying the values for the indirect dimension. The optional argument `-phsens` assumes phase-sensitive 2D data.

`fbc` *d order* {{*from to*} ...} [*skip*]
Baseline corrects the data set *d* by fitting every `skip` (default 1) data point of the baseline in the defined frequency ranges to a polynomial of order *order*.

`fnewnp` *d points*
Changes the number of data points in the data set *d*. Intermediate points are interpolated using a cubic spline.

`fsmooth` *d points order*
Smoothing of the data set *d* to a given order *order*.

TABLE 2—*Continued*

peaks ffindpeaks *d th sens* [*from to*]
Finds all peaks in the data set *d* that are higher than *th* and spans at least *sens* data points (optionally restricted to searching inside a specific frequency range) and returns a list of frequencies and peak heights.

areas fint *d* {{*from1 to1*} {*from2 to2*}...}
Returns a list of integrated intensities (by summation) for the specified spectral regions in the data set *d*.

areas fssbint *d dny shift width*
Returns a list of integrated intensities (by summation) for equidistant spectral regions (separated by *dny* Hz, centered around *shift* Hz, and each having a width of *width*) in the data set *d*.

d fdup *s*
Copies the data set *s* to a new dataset *d*.

fcopy *d s*
Copies the data set *s* into an existing data set *d*.

fadd *d s*
Adds the data sets *s* and *d* and saves the result in *d*.

fsub *d s*
Subtracts the data set *s* from *d* and saves the result in *d*.

frev *d*
Reverses the order of all data points in the data set *d*.

rms frms *d1 d2* [-re | -im] [{{*from to*}...}]
Returns the normalized root-means-square deviation between the complex, real, or imaginary part of two data sets *d1* and *d2,* optionally within specific frequency ranges.

fextract *d from to*
Shrinks the data set *d* to the specified frequency range.

d fzero [{{*from to*} ...}]
Attributes zero intensity to frequency regions (or the full region) of the data set *d*.

areas faddpeaks *d cutoff* {{*frq int lb r*} ... }
Adds a series of peaks to the data set *d*, each of which is specified by a frequency, intensity, additional linebroadening, and Gauss/Lorentz ratio. The areas of the peaks are returned as a list. The *cutoff* parameter defines the minimum intensity calculated before it is truncated to zero.

fexpr *d reexpr imexpr*
Applies Tcl expressions to the real and imaginary part of each data point in the data set *d*. Valid variables are the real part of the complex data point $re, the imaginary part $im, and the point index $i starting from one. These variables must be preceded with a backslash if a local variable (fac) is used, e.g., fexpr $f [list \$re*$fac] {$im+$i*1.23}.

v findex *d i* [-re | -im]
Returns the real and/or the imaginary part of the *i*th complex data point in a data set *d*.

fsetindex *d i re im*
Sets the real and imaginary part of the *i*th complex data point in the data set *d*.

v fx *d i*
Returns the frequency or time of a data point *i* depending on the type of the data set *d*.

d fload *file*
Loads the data set *d* from a *file* and returns a data descriptor.

funload [*d*]
Removes all or a specific data set *d* from the memory.

d fcreate -np *v* -sw *v* [-ref *v* -ni *v* -sw1 *v* -ref1 *v* -type *v*]
Creates and returns a descriptor to the new data set *d* with zero points and with specifications corresponding the arguments of which -np and -sw are required and *v* for -type is either fid or spe.

v fget *d* [-ref | -ref1 | -sw | -sw1 | -np | -ni | -type]
Returns either the reference line, spectral width, number of complex data points, or type of data (fid or spe) from the data set *d* depending on the argument.

fset *d* [-ref *v* -ref1 *v* -sw *v* -sw1 *v* -type *v*]
Changes the specifications for the data set *d* following the syntax from fcreate.

flt *array*
Performs iterative fitting using parameters given in the array *array* as described in the text.

fplot2d *d name* (-ppm | -ps) [*scale*]
Creates a Postscript (-ps) or portable pixmap (-ppm) bitmap plot of a 2D data set *d* using conventional 2D plotting conventions, i.e., shift increasing left and down, optionally vertically scaled with *scale*.

r_{12} dip2dist N_1 N_2 $b_{12}/2\pi$
Calculates the distance (in Å) between the nuclei N_1 and N_2 based on the dipolar coupling constant $b_{12}/2\pi$.

$b_{12}/2\pi$ dist2dip N_1 N_2 r_{12}
Analogues to dip2dist but calculates the dipolar coupling constant in hertz from the distance r_{12} (in Å).

list csapar δ_{11} δ_{22} δ_{33}
Returns the isotropic shift, chemical shift anisotropy, and the asymmetry parameter assuming unordered principal elements (in hertz or ppm) as arguments.

TABLE 2—*Continued*

list `csaprinc` δ_{iso} δ_{aniso} η
Returns the ordered principal elements δ_{xx}, δ_{yy}, and δ_{zz} given the isotropic shift, chemical shift anisotropy, and the asymmetry parameter (in hertz or ppm) as arguments.

list `isotopes`
Returns a list with data for the spin isotopes available.

value `gamma` *N*
Returns the magnetogyric ratio of a nucleus *N* given in the unit 10^7rad/(*Ts*).

value `resfreq` *N* [$|\omega_0/2\pi|$]
Returns the value of the absolute resonance frequency in hertz for at nucleus *N* assuming an absolute proton resonance frequency of 10^6 Hz (default) or optionally $|\omega_0/2\pi|$ Hz.

Tcl language constructs

name arg arg ...	Function call with arguments.
`set` *var value*	Sets a variable to a value.
array(*var*)	A variable in an associative array.
{ ... }	Begin and end of a command block.
\	Continuation of a line.
$*var*	Gets a value of a variable.
[`expr` *expression*]	Evaluates a mathematical expression.
`proc` *name* {*args*} {*body*}	Definition of a user procedure.
`global` *var var ...*	Make variables visible outside the current block.
`for` {*start*} {*test*} {*incr*} {*body*}	A `for` loop.
`if` {*test*} {*body*} `elseif` {*test*} {*body*} `else` {*test*} {*body*}	An `if` construct.
[`list` e_1 e_2 ...]	Creates a list with elements e_i.
[`lindex` $*list i*]	Returns element *i* from a list (counting from zero).

[a] All values are in hertz, microseconds, and degrees if not specified otherwise. Arguments in square brackets are optional.
[b] The parameter can be given in ppm (δ value) by appending a `p` to the value.
[c] The *order* parameter for the quadrupolar interaction can be 1 or 2 corresponding to first and second order, respectively, according to Eq. [10].
[d] For half-integer quadrupolar nuclei I*ic* may be applied to excite and detect only the {1/2, −1/2} central transition.