.BGSD File Syntax Documentation

The purpose of the BGSD file is to contain all information about a specific fully causal bond graph system, including all of its elements and their constituent equations or values, initial conditions, and bonds which connect the elements. Various runtime options, primarily useful for the purpose of analysis, can also be specified in this file.

BGSD stands for Bond Graph System Descriptor. The BGSD file is referred to as "the BGSD" in this documentation.

BGSolver v1.03 is the MATLAB code package used to construct and process BGSD files. It is referred to as "the software" in parts of this documentation. BG Processor (referred to as "the processor" in this documentation) is the part of BGSolver v1.03 responsible for reading the BGSD file, formulating and sorting the algebraic equations, extracting and integrating the state derivative vector, and, optionally, evaluating some or all of the bond variables in the problem at the time points returned by the time integrator.

BGSD format differs between versions of the code, because different versions of BGSolver support different types of elements and expressions. The format provided here is for code version 1.03.

The format presented in this file is not problem-specific, and can be used for describing any BGS, modeling any physical system. Any fully causal BGS described by this format can be processed by BGSolver v1.03.

Some lines in this specification are part of the version 1.03 .BGSD file format, but are not supported by BGSolver v1.03. Such lines have **bold red restrictions** in their description in the documentation.

General notes

A BGSD file consists of 11 sections, in this order:

1. File Header Section
2. Runtime and Recording Options Section
3. Time Integrator Options Section
4. Solver and Jumpstart Options
5. Evaluator Options Section
6. Post-Processor Options Section
7. Linked Files and Directories Section
8. Additional System Information Section
9. Bond Connectivity Map Section
10. Element List Section.
11. File End Section

These sections can be separated into 2 major categories:

- Run Options – sections 1, 2, 3, 4, 5, 6 and 7.
- Bond Graph System – sections 8, 9 and 10.

Section 11 does not contain any information. All of the options in the Run Options sections can be overridden by the options that the processor is called with. None of the system information in the Bond Graph System sections can be overridden by the processor, with the exception of initial conditions, state variables' non-negativity statuses and absolute tolerances. All of these modifications to the Bond Graph System can only be made by restarting the processor for a previously sorted system; they cannot be made during the first processing (including sorting) of the system.

Certain options have default values – that means that when the file is written using the standard tools for it, if the corresponding option is not specified, its default value will be recorded.

The BGSD file is an ASCII file which consists of lines. The general line format is presented below. Notice that only text in `Courier font` appears in the BGSD, the rest is just comments.

Line format

A BGSD file consists of lines. Almost any line has the following general syntax:

```
keyword string
```

Here, the keyword indicates what type of information (if any) is contained in the string. The keyword cannot be empty, but the string can be. They are separated by exactly one space.

Notice, that the keyword appears exactly as shown in the text below, while the string is different based on whatever information that string is intended to contain. For example, if the BGSD file is intended for software version 1.03, the following will be the `SVF` line in the File Header:

```
SVF 1.03
```

The 11 sections of the BGSD file are presented below. In every section description, the syntax of the section is given first, followed by an explanation of the meaning of each line.

1. File Header

```
BGSD
SVF Version
SVC Version
NOTES If_Notes
Notes_Entry (Optional)
NOTESEND
```

Comments

I. `BGSD` – "Bond Graph System Descriptor" keyword. It is the first line of the file header.

II. `SVF` – "Software Version For" keyword.
`Version` – The version number of the processing software that the BGSD was created for.

III. `SVC` – "Software Version Created" keyword.
`Version` – The version of the software that the BGSD was created by. If `SVC` and `SVF` mismatch, the processing software throws a warning. By default, `SVC` and `SVF` are the same.

IV. `NOTES` – "Notes" keyword. It indicates the beginning of the optional Notes section.
`If_Notes` – Whether or not there are any optional notes entered. It can be one of the following keywords:
- `0` – There are no optional notes entered. This is the default option.
- `1` – There are optional notes entered, in the following line.

V. `Notes_Entry` – The optional notes entry. This is a line of any text, in double quotes. (This line is only present with `NOTES` of `1`).

VI. `NOTESEND` – "Notes End" keyword. It's present whether or not any optional notes were entered.

2. Runtime and Recording Options

RRO
VERBOSE How_Verbose
RTI If_Run_TI
LOG How_Log (*Optional*)
LOGFNSPEC Log_FName_Spec (*Optional*)
LOGFN Log_FName (*Optional*)
SMR If_SrtMat_Rec
SMFNSPEC SrtMat_FName_Spec (*Optional*)
SMFN SrtMat_FName (*Optional*)
RTRR If_RTR_Rec
RTRRFQSPEC RTR_Rec_Freq_Spec (*Optional*)
RTRRFQ RTR_Rec_Freq (*Optional*)
RTRRFNSPEC RTR_Rec_FName_Spec (*Optional*)
RTRRFN RTR_Rec_FName (*Optional*)
RRM If_Res_Rec_Mat
RRMFNSPEC Res_Rec_Mat_FName_Spec (*Optional*)
RRMFN Res_Rec_Mat_FName (*Optional*)
RROEND

Comments

I. RRO – "Runtime and Recording Options" keyword.

II. VERBOSE – "Verbose" keyword.
How_Verbose – How verbose the BGSolver must be when processing the file. How_Verbose can be one of the following keywords:
- 0 – The processor is silent, and does not report any steps when processing the file. This setting is equivalent to calling the processor with the "silent" option on, and is therefore incompatible with any settings that request input from the user.
- 1 – The processor is not silent or verbose, and only reports major steps when processing the file. This is the default option.
- 2 – The processor is verbose, and reports both major and minor steps when processing the file.

III. RTI – "Run Time Integrator" keyword.
If_Run_TI – Whether or not to run the time integrator, or stop after the AEs are sorted. If_Run_TI can be one of the following keywords:
- 0 – Stop the processing after the AEs are sorted. If this option is chosen, it is recommended to save the sorted expressions and supporting functions by choosing an appropriate option in the SMR line, to be able to later run a time integrator. Note, that time integration settings (time integrator options, runtime results recording options, solver and jumpstart options, etc.) still must be specified, although they will not be used if this option is chosen.
- 1 – Run the time integrator after the AEs are sorted. This is the default option.

IV. LOG – "Log" keyword. (This line is only present with VERBOSE other than 0).

`How_Log` – How to log all the outputs specified by `VERBOSE`. `How_Log` can be one of the following keywords:

- `0` – Report all specified outputs in the MATLAB command line. This is the default option.
- `1` – Record all specified outputs in a log file.
- `2` – Both report all specified outputs in the MATLAB command line and record them in a log file.

V. `LOGFNSPEC` – "Log File Name Specification" keyword. (This line is only present with `VERBOSE` other than `0` and `LOG` other than `0`).

`Log_FName_Spec` – How log file name is specified. `Log_FName_Spec` can be one of the following keywords:

- `0` – Use the default log file name `<(Insert BGSD File Name here).log>`. This file is placed in the `<\Results>` subdirectory of the directory the BGSD file is in. This is the default option.
- `1` – The log file name is specified in the next line.
- `2` – Request the log file name during processing. If the processor was called with the "silent" option on, this option cannot be used.

VI. `LOGFN` – "Log File Name" keyword. (This line is only present with `LOGFNSPEC` of `1`).

`Log_FName` – File name of the log file, in double quotes. Either a full file path is specified, or only the file name. If only the file name is specified, the log file is saved into the `<\Results>` subdirectory of the directory the BGSD file is in.

VII. `SMR` – "Sorted .mat file Record" keyword.

`If_SrtMat_Rec` – Whether or not to record the sorted expressions and supporting functions and variables into a .mat file after the AEs are sorted. This happens at the end of Bond Graph Step 6. `If_SrtMat_Rec` can be one of the following keywords:

- `0` – Do not record the sorted expressions into a .mat file.
- `1` – Record the sorted expressions into a .mat file. This is the default option.

VIII. `SMFNSPEC` – "Sorted .mat File Name Specification" keyword. (This line is only present with `SMR` of `1`).

`SrtMat_FName_Spec` – How sorted .mat file name is specified. `SrtMat_FName_Spec` can be one of the following keywords:

- `0` – Use the default sorted .mat file name `<(Insert BGSD File Name here)_SrtRes.mat>`. This file is placed in the `<\Results>` subdirectory of the directory the BGSD file is in. This is the default option.
- `1` – The sorted .mat file name is specified in the next line.
- `2` – Request the sorted .mat file name during processing. If the processor was called with the "silent" option on, this option cannot be used.

IX. `SMFN` – "Sorted .mat File Name" keyword. (This line is only present with `SMR` of `1` and `SMFNSPEC` of `1`).

`SrtMat_FName` – .mat file name into which the sorted expressions and supporting functions are saved after the AEs are sorted, in double quotes. Either a full file path is specified, or only the file name. If only the file name is specified, the .mat file is saved into the `<\Results>` subdirectory of the directory the BGSD file is in.

X. `RTRR` – "Runtime Results Recording" keyword.

`If_RTR_Rec` – Whether or not to record the integration results into a text file as they are being computed. `If_RTR_Rec` can be one of the following keywords:

- `0` – Do not record the runtime integration results into a text file. This is the default option.
- `1` – Record the runtime integration results into a text file. This option is only possible if fixed time stepping (either with or without sequential adaptive time integrations) is specified in `TST` line in the Time Integrator Options section.

XI. `RTRRFQSPEC` – "Runtime Results Recording Frequency Specification" keyword. (This line is only present with `RTRR` of `1`).

`RTR_Rec_Freq_Spec` – How the runtime results recording frequency is specified. `RTR_Rec_Freq_Spec` can be one of the following keywords:

- `1` – The recording frequency is specified in the next line.
- `2` – Request the recording frequency during processing. If the processor was called with the "silent" option on, this option cannot be used. This is the default option.

XII. `RTRRFQ` – "Runtime Results Recording Frequency" keyword. (This line is only present with `RTRR` of `1` and `RTRRFQSPEC` of `1`).

`RTR_Rec_Freq` – Frequency with which to record the integration results into the text file, expressed as the integer number of fixed time steps (specified in `FTS` line in the Time Integrator Options section) to take between each recording. This number has to be greater than or equal to one.

XIII. `RTRRFNSPEC` – "Runtime Results Record File Name Specification" keyword. (This line is only present with `RTRR` of `1`).

`RTR_Rec_FName_Spec` – How runtime results record file name is specified. `RTR_Rec_FName_Spec` can be one of the following keywords:

- `0` – Use the default runtime results record file name `<(Insert BGSD File Name here)_RTRes.txt>`. This file is placed in the `<\Results>` subdirectory of the directory the BGSD file is in. This is the default option.
- `1` – The runtime results record file name is specified in the next line.
- `2` – Request the runtime results record file name during processing. If the processor was called with the "silent" option on, this option cannot be used.

XIV. `RTRRFN` – "Runtime Results Record File Name" keyword. (This line is only present with `RTRR` of `1` and `RTRRFNSPEC` of `1`).

`RTR_Rec_FName` – Text file name into which the runtime results are written, in double quotes. This file is an ASCII text file, with a .txt extension. Either a full file path is specified, or only the file name. If only the file name is specified, the file is saved into the `<\Results>` subdirectory of the directory the BGSD file is in.

XV. `RRM` – "Results Record .mat file" keyword.

`If_Res_Rec_Mat` – Whether or not to record the final results in a .mat file. The final results include the integrated state vectors, the post-processing results and the time points vector. `If_Res_Rec_Mat` can be one of the following keywords:

- `0` – Do not record the final results in a .mat file.
- `1` – Record the final results in a .mat file. This is the default option.

XVI. RRMFNSPEC – "Results Record .mat File Name Specification" keyword. (This line is only present with RRM of 1).
Res_Rec_Mat_FName_Spec – How results record .mat file name is specified. Res_Rec_Mat_FName_Spec can be one of the following keywords:
- 0 – Use the default results record .mat file name <(Insert BGSD File Name here)_FinRes.mat>. This file is placed in the <\Results> subdirectory of the directory the BGSD file is in. This is the default option.
- 1 – The results record .mat file name is specified in the next line.
- 2 – Request the results record .mat file name during processing. If the processor was called with the "silent" option on, this option cannot be used.

XVII. RRMFN – "Results Record .mat File Name" keyword. (This line is only present with RRM of 1 and RRMFNSPEC of 1).
Res_Rec_Mat_FName – .mat file name into which the final results are saved, in double quotes. Either a full file path is specified, or only the file name. If only the file name is specified, the .mat file is saved into the <\Results> subdirectory of the directory the BGSD file is in.

XVIII. RROEND – "Runtime and Recording Options End" keyword.

3. Time Integrator Options

```
TIO
TITSPEC TIT_Spec
TIT TIT_Info (Optional)
IISPEC How_II_Spec
II T0 TF (Optional)
TST TS_Type
FTSSPECT FTS_Spec_Type (Optional)
FTSSPEC FTS_Spec (Optional)
FTS Time_Step (Optional)
PHXSPLIT Physics_Split
ADDTIO Addtnl_TIO_Num
ADDTIO_Pairs_List (Optional)
TIDISP TI_Display
TIOEND
```

Comments

I. TIO – "Time Integrator Options" keyword.

II. TITSPEC – "Time Integrator Type Specification" keyword.
TIT_Spec – How the time integrator type is specified in the BGSD file. TIT_Spec can be one of the following keywords:
- 1 – The time integrator type is specified in the file, in the following line. This is the default option.
- 2 – Request the time integrator type during the processing. If the processor was called with the "silent" option on, this option cannot be used.

III. TIT – "Time Integrator Type" keyword. (This line is only present with TITSPEC of 1).

`TIT_Info` – Type of time integrator to use. `TIT_Info` can be one of the following keywords:

- `ode15s` – Use MATLAB's `ode15s` solver (stiff systems, medium accuracy). This is the default solver.
- `ode45` – Use MATLAB's `ode45` solver (non-stiff systems, medium accuracy).
- `ode113` – Use MATLAB's `ode113` solver (non-stiff systems, high accuracy).
- `ode23` – Use MATLAB's `ode23` solver (moderately stiff systems, low accuracy).
- `ode23s` – Use MATLAB's `ode23s` solver (stiff systems, low accuracy).
- `ode23t` – Use MATLAB's `ode23t` solver (moderately stiff systems, low accuracy, no numerical damping).
- `ode23tb` – Use MATLAB's `ode23tb` solver (stiff systems, low accuracy).
- `custom` – .m-file name (not full path) of a manually-written time integrator with an appropriate signature (primarily for convergence studies). The function must be either in MATLAB's default path, in one of the directories specified in the Linked Files and Directories section, or in BGSolver's `<\Time_Integrators>` subdirectory. Such function must accept the same type of basic arguments (function handle, time interval, initial guess) as the MATLAB ODE solvers. Split-supporting and fixed-step time integrators accept additional data as fields in their ODE Options argument (ODE Options as constructed by MATLAB's `odeset`, with additional manually-added fields).

IV. `IISPEC` – "Integration Interval Specification" keyword.

`II_Spec` – How the integration interval is specified. `II_Spec` can be one of the following keywords:

- `0` – Integration interval is specified in the file, in the following line.
- `1` – Request the integration interval during processing. If the processor was called with the "silent" option on, this option cannot be used. This is the default option.

V. `II` – "Integration Interval" keyword. (This line is only present with `IISPEC` of `0`).

`T0` – Beginning of integration interval.

`TF` – End of integration interval.

VI. `TST` – "Time Step Type" keyword.

`TS_Type` – Whether or not the time step type is adaptive or fixed. Note, that while an adaptive time step requires the use of an adaptive time integrator, a fixed time step can be used with both an adaptive and a fixed time integrator. If a fixed time step is used with an adaptive time integrator (one of the options), the internal time stepping within a given (fixed) time step is always adaptive. However, in this case, the returned results are evaluated at the specified (equally stepped) time points, using sequentially executed adaptive time integrations. `TS_Type` can be one of the following keywords:

- `0` – Adaptive time-stepping is used. This option can only be used if an adaptive time integrator is specified in the `TIT` line. This is the default option.
- `1` – Fixed time-stepping is used. This option can be only used if a fixed time integrator is specified in the `TIT` line.
- `2` – Fixed time-stepping between sequentially executed adaptive time integrations is used. This option can only be used if an adaptive time integrator is specified in the `TIT` line.

Only options `0` and `1` are currently supported!

VII. `FTSSPECT` – "Fixed Time Step Specification Type" keyword. (This line is only present with `TST` of `1` or `2`).

`FTS_Spec_Type` – What the fixed time step is specified in terms of. `FTS_Spec_Type` can be one of the following keywords:

- `0` – The integer number of equal fixed time steps in the integration interval is specified. This number is one greater than the resulting number of computed states, including the initial state. This is the default option.
- `1` – The size of one fixed time step is specified. To fully integrate over the specified integration interval, this time step size must fit an integer number of times into the integration interval.

VIII. `FTSSPEC` – "Fixed Time Step Specification" keyword. (This line is only present with `TST` of `1` or `2`).

`FTS_Spec` – How the fixed time step is specified. `FTS_Spec` can be one of the following keywords:

- `0` – The fixed time step is specified in the file, in the following line.
- `1` – Request the fixed time step during processing. If the processor was called with the "silent" option on, this option cannot be used. This is the default option.

IX. `FTS` – "Fixed Time Step" keyword. (This line is only present with `TST` of `1` or `2` and `FTSSPEC` of `0`).

`Time_Step` – Time step size, specified according to `FTSSPECT`.

X. `PHXSPLIT` – "Physics Split" keyword.

`Physics_Split` – Whether, and how, the physics are split. `Physics_Split` can be one of the following keywords:

- `0` – Full coupling. The physics are not split. Any time integrator will work with this option. This is the default option.
- `1` – Sequential (triangular) coupling. The physics are split and integrated according to the specified physics order. After a given physics type is integrated, the new time step's value of this physics is used in all higher-level physics integrations.
- `2` – Full operator split (diagonal coupling). The physics are completely split, and integrated separately. For the integration of any single physics, all other physics' values are taken from the previous time step, including the lower-level physics.

Options `1` and `2` can only be used with split-supporting integrators specified in the `TIT` line.

BGSolver v1.03 does not currently come with any custom time integrators, and built-in MATLAB time integrators do not support physics splitting. Therefore, only option `0` is currently supported, unless the user chooses to implement a custom split-supporting time integrator.

XI. `ADDTIO` – "Additional Time Integrator Options" keyword.

`Addtnl_TIO_Num` – Number of additional time integrator options pairs listed below. If `0`, no additional time integrator options are listed below.

XII. `ADDTIO_Pairs_List` – List of (parameter name, "parameter value") pairs, one pair per line, of the additional time integrator options. These parameters are normally the same as the ones specified in MATLAB's built-in `odeset` function arguments. The parameter values are listed in double quotes, and are executed (with `eval`) prior to being submitted as time integrator options, therefore they can contain references to variables or objects

contained in Linked Files and Directories lists. If a parameter must stay a string after having `eval` executed on it, the parameter must have single quotation marks placed around it, in addition to the double quotation marks. This list is only present if `ADDTIO` is specified as greater than zero.

XIII. `TIDISP` – "Time Integrator Display" keyword.

`TI_Display` – Time integrator level of display. This option is only utilized for MATLAB built-in time integrators, that can make use of the `OutputFcn` parameter. Time stepwise output of the custom time integrators can be configured through the `SDISP` field in the Solver and Jumpstart Options section. Can be one of the following keywords:

- `0` – No time stepwise time integrator display updates are output.
- `1` – Some statistical information is output at every time step.

All outputs specified by this option are output only to the command window, regardless of the specified processor verbosity.

XIV. `TIOEND` – "Time Integrator Options End" keyword.

4. Solver and Jumpstart Options

`SJSO`
`STOLX Sol_Tol_X`
`STOLRES Sol_Tol_Res`
`SDISP Sol_Display`
`SRSCAL Sol_Res_Scal`
`SXSCAL Sol_X_Scal`
`SXCENT Sol_X_Cent`
`SPC Sol_PreCon`
`SIGPDCT Sol_IG_Predict`
`SJACOB Sol_Jacobian`
`SJSP Sol_Jac_Sparsity`
`ADDSO Addtnl_SO_Num`
`ADDSO_Pairs_List` (*Optional*)
`JSTYPE JStart_Type`
`JSMSNUM JStart_Min_Step_Num`
`JSPHXSPLIT JStart_Physics_Split`
`JSTOLX JStart_Tol_X`
`JSTOLRES JStart_Tol_Res`
`JSTIT JStart_TI_Info`
`JSDISP JStart_Display`
`JSRSCAL JStart_Res_Scal`
`JSXSCAL JStart_X_Scal`
`JSXCENT JStart_X_Cent`
`JSPC JStart_PreCon`
`JSJACOB Jstart_Jacobian`
`JSJSP JStart_Jac_Sparsity`
`ADDJSO Addtnl_JSO_Num`
`ADDJSO_Pairs_List` (*Optional*)
`SJSOEND`

Comments

This section specifies the options for the custom time integrators' nonlinear solvers and their jumpstart procedures. Default values must be entered in the required lines if a built-in MATLAB time integrator (like `ode15s`) is used. BGSolver v1.03 currently does not come with any custom time integrators, and so relies on the MATLAB ODE integration suite for integrating the state derivative vector. The MATLAB built-in time integrators do not use any of the options in this section, and so this section is primarily available for users to control their own custom time integrators, if they choose to develop any.

Most of the options available in this section expect MATLAB Optimization Toolbox's `fsolve` to be the nonlinear solver used by the implicit time integrator, or the jumpstarter, to take the time steps. Some of the options, such as scaling, imply an explicit modification of the residual vector function by the time integrator prior to supplying it to the nonlinear solver. If a custom time integrator does not support one or more of such options, it is reasonable to leave them default in the BGSD file, and simply ignore them when supplying the time integrator options (if any) to the custom time integrator.

"Initial guess," in context of this section, is the previous time step's state vector, which is used as the initial guess for the nonlinear solver when solving for the new time step's state vector.

I. `SJSO` – "Solver and Jumpstart Options" keyword.

II. `STOLX` – "Solver Tolerance on $\vec{\mathbf{x}}$" keyword.

`Sol_Tol_X` – Solver tolerance on the state vector value. It can be one of the following values:

- `0` – Either solver tolerance on $\vec{\mathbf{x}}$ is not needed by the specified time integrator, or use the default tolerance (set by the specified time integrator; MATLAB `fsolve` default is 10^{-6}). This is the default option.
- `number` – A positive number to use as the solver tolerance on $\vec{\mathbf{x}}$.

III. `STOLRES` – "Solver Tolerance on Residual" keyword.

`Sol_Tol_Res` – Solver tolerance on the residual (function being minimized to zero by the solver). It can be one of the following values:

- `0` – Either solver tolerance on the residual is not needed by the specified time integrator, or use default tolerance (set by the specified time integrator; MATLAB's default is 10^{-6}). This is the default option.
- `number` – A positive number to use as the solver tolerance on the residual.

IV. `SDISP` – "Solver Display" keyword.

`Sol_Display` – Solver level of display, as specified in MATLAB nonlinear solver specification. It can be one of the following values (all corresponding to specific MATLAB nonlinear solver level of display settings):

- `-1` – Use the solver display level specified by the time integrator. This is the default option.
- `0` – Solver display level set to "`off`". Absolutely no output is provided, whether or not the nonlinear solver succeeds.
- `1` – Solver display level set to "`iter`". Output is displayed at every iteration.
- `2` – Solver display level set to "`final`". Output is displayed only at final iteration.

- `3` – Solver display level set to "`notify`". Output is displayed only if the nonlinear solver fails.

For all options other than `0` (and `-1`, if the time integrator specifies a display level of `0`), the time value is output prior to the solver output. Note, that all these outputs are provided to the command window, regardless of the specified processor verbosity.

V. `SRSCAL` – "Solver Residual Scaling" keyword.

`Sol_Res_Scal` – Type of residual function scaling to use before running the solver at every time step (the solver itself may implement additional scaling operations, separate from these options; below only the pre-solver scaling is considered). It can be one of the following keywords:

- `-1` – Use the solver residual scaling specified by the time integrator, if any. This is the default option.
- `0` – Do not use any solver residual scaling.
- `1` – Use initial guess residual vector-based solver residual scaling.

VI. `SXSCAL` – "Solver $\vec{\mathbf{x}}$ Scaling" keyword.

`Sol_X_Scal` – Type of state vector scaling to use before running the solver at every time step (both in-solver and pre-solver scaling is considered below). It can be one of the following keywords:

- `-1` – Use the state vector scaling specified by the time integrator, if any. This is the default option.
- `0` – Do not use any state vector scaling.
- `1` – Use pre-solver initial guess-based state vector scaling.
- `2` – Use initial guess-based `TypicalX` solver option.
- `3` – Use Jacobian column sum-based state vector scaling. This type of scaling will require the construction of the Jacobian matrix based on initial guess state vector.

VII. `SXCENT` – "Solver $\vec{\mathbf{x}}$ Centering" keyword.

`Sol_X_Cent` – Type of state vector centering to use before running the solver at every time step (the solver itself may implement additional centering operations, separate from these options; below only the pre-solver scaling is considered). It can be one of the following keywords:

- `-1` –Use the state vector centering specified by the time integrator, if any. This is the default option.
- `0` – Do not use any state vector centering.
- `1` – Use initial guess-based state vector centering, with centering about the zero vector.
- `2` – Use initial guess-based state vector centering, with centering about the unity vector.

VIII. `SPC` – "Solver Preconditioner" keyword.

`Sol_PreCon` – Type of residual function preconditioner to use before running the solver at every time step (the solver itself may implement additional preconditioning operations, separate from these options; below only the pre-solver preconditioning is considered). It can be one of the following keywords:

- `-1` – Use the preconditioning specified by the time integrator, if any. This is the default option.
- `0` – Do not use any preconditioning.

- `1` – Use Jacobian ILU preconditioner. This preconditioner will require the construction of the Jacobian matrix based on initial guess state vector.

IX. `SIGPDCT` – "Solver Initial Guess Predictor" keyword.

`Sol_IG_Predict` – Whether or not to use a low-order time integrator to obtain a better initial guess state vector for the solver. The same pre-solver preconditioner and scaling options specified for the solver are used for this predictor, but they all use the previous time step's state vector as the initial guess. It can be one of the following keywords:

- `-1` – Use the initial guess predictor specified by the time integrator, if any. This is the default option.
- `0` – Do not use an initial guess predictor.
- `1` – Use 1st-order backward Euler time integrator to take a time step, and use it as an initial guess.

X. `SJACOB` – "Solver Jacobian" keyword.

`Sol_Jacobian` – Whether, and how, an externally-constructed Jacobian is supplied to the solver. It can be one of the following keywords:

- `-1` – Use the solver Jacobian setting specified by the time integrator, if any. This is the default option.
- `0` – Do not supply an externally-constructed Jacobian to the solver.
- `1` – Construct a Jacobian function based on MATLAB's built-in `numjac` function to pass to the solver. State absolute tolerance is used as threshold values vector. No typical state vector is provided.
- `2` – Construct a Jacobian function based on MATLAB's built-in `numjac` function to pass to the solver. State absolute tolerance is used as threshold values vector. Initial guess is provided as the typical state vector.

XI. `SJSP` – "Solver Jacobian Sparsity" keyword.

`Sol_Jac_Sparsity` – Whether, and how, the Jacobian sparsity is treated. It can be one of the following keywords:

- `-1` – Jacobian sparsity treatment is specified by the time integrator, if any. This is the default option.
- `0` – Jacobian is treated as a full matrix.
- `1` – Jacobian is treated as a sparse matrix. The sparsity pattern is constructed exactly during sorting.
- `2` – Jacobian is treated as a sparse matrix. The sparsity pattern is obtained by using MATLAB's built-in `numjac` function on the initial state vector. This sparsity pattern is assumed to stay constant throughout the run. It is often incomplete, so option `1` is preferred.

Currently, the Jacobian sparsity pattern is always constructed exactly during sorting; it is supplied to the built-in MATLAB time integrators.

XII. `ADDSO` – "Additional Solver Options" keyword.

`Addtnl_SO_Num` – Number of additional solver options pairs listed below. If `0`, no additional solver options are listed below.

XIII. `ADDSO_Pairs_List` – List of (parameter name, "parameter value") pairs, one pair per line, of additional solver options. These parameters are normally the same as the ones specified in MATLAB's built-in `optimset` function arguments. The parameter values are

listed in double quotes, and are executed (with `eval`) prior to being submitted as solver options, therefore they can contain references to variables or objects contained in Linked Files and Directories lists. If a parameter must stay a string after having `eval` executed on it, the parameter must have single quotation marks placed around it, in addition to the double quotation marks. This list is only present if `ADDSO` is specified as greater than zero.

XIV. `JSTYPE` – "Jumpstart Type" keyword.

`JStart_Type` – Type of jumpstarter to use, if required by the chosen time integrator. It can be one of the following values:

- `-1` – Jumpstarter type specified by the time integrator, if any. This is the default option.
- `0` – If n consecutive time steps are required by the time integrator, use the following: $\vec{\mathbf{x}}^{0} = \vec{\mathbf{x}}^{-1} = \ldots = \vec{\mathbf{x}}^{1-n}$.
- `1` – Use a ministep-based jumpstarter, configured according to the options below.

XV. `JSMSNUM` – "Jumpstart Ministep Number" keyword.

`JStart_Min_Step_Num` – Number of ministeps the jumpstarter should take per 1 full time step. It can be one of the following values:

- `0` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use default number of ministeps (set by the specified time integrator; for most custom time integrators this is 100). This is the default option.
- `number` – A positive number of ministeps to take per 1 full time step.

XVI. `JSPHXSPLIT` – "Jumpstart Physics Split" keyword.

`JStart_Physics_Split` – Whether, and how, the physics are split during jumpstart. It can be one of the following keywords:

- `-1` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use default physics split option during jumpstart (set by the specified time integrator; for most custom time integrators jumpstart integration is fully coupled). This is the default option.
- `0`, `1`, or `2` – Use this physics split option during jumpstart. The meanings of each option are the same as in the `PHXSPLIT` line.

XVII. `JSTOLX` – "Jumpstart Tolerance on $\vec{\mathbf{x}}$" keyword.

`JStart_Tol_X` – Solver tolerance on the state vector value to use during jumpstart. It can be one of the following values:

- `0` – Either ministep-based jumpstart is not used or required by the specified time integrator, solver tolerance on $\vec{\mathbf{x}}$ is not needed by the specified time integrator's jumpstarter, or use the default tolerance (set by the specified time integrator; MATLAB's default is 10^{-6}). This is the default option.
- `number` – A positive number to use as the solver tolerance on $\vec{\mathbf{x}}$ during jumpstart.

XVIII. `JSTOLRES` – "Jumpstart Tolerance on Residual" keyword.

`JStart_Tol_Res` – Solver tolerance on the residual (function being minimized to zero by the solver) to use during jumpstart. It can be one of the following values:

- `0` – Either ministep-based jumpstart is not used or required by the specified time integrator, solver tolerance on the residual is not needed by the specified time integrator's jumpstarter, or use default tolerance (set by the specified time integrator; MATLAB's default is 10^{-6}). This is the default option.

- `number` – A positive number to use as the solver tolerance on the residual during jumpstart.

XIX. `JSTIT` – "Jumpstart Time Integrator Type" keyword.

`JStart_TI_Info` – Time integrator type to use during jumpstart. It can be one of the following keywords:

- `default` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use default jumpstart time integrator (set by the specified time integrator; for most custom time integrators, this is 1st-order backward Euler). This is the default option.
- `bkwdEuler1` – Use 1st-order backward Euler time integrator during jumpstart.

Note, that unlike in the `TIT` line, the `JSTIT` is a string setting, and not the name of a time integrator .m file.

XX. `JSDISP` – "Jumpstart Solver Display" keyword.

`JStart_Display` – Solver level of display to use during jumpstart, as specified in MATLAB nonlinear solver specification. It can be one of the following values (all corresponding to specific MATLAB nonlinear solver level of display settings):

- `0` – Use the jumpstart solver display level specified by the time integrator. This is the default option.
- `1` – Jumpstart solver display level set to "`off`". Absolutely no output is provided, whether or not the nonlinear solver succeeds.
- `2` – Jumpstart solver display level set to "`iter`". Output is displayed at every iteration.
- `3` – Jumpstart solver display level set to "`final`". Output is displayed only at final iteration.
- `4` – Jumpstart solver display level set to "`notify`". Output is displayed only if the nonlinear solver fails.

For all options other than `0` (and `-1`, if the time integrator specifies a display level of `0`), the time value is output prior to the solver output. Note, that all these outputs are provided to the command window, regardless of the specified processor verbosity.

XXI. `JSRSCAL` – "Jumpstart Residual Scaling" keyword.

`JStart_Res_Scal` – Type of residual function scaling to use before running the jumpstart solver at every time ministep (the solver itself may implement additional scaling operations, separate from these options; below only the pre-solver scaling is considered). It can be one of the following keywords:

- `-1` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use the jumpstart solver residual scaling specified by the time integrator, if any. This is the default option.
- `0` – Do not use any jumpstart solver residual scaling.
- `1` – Use initial guess residual vector-based jumpstart solver residual scaling.

XXII. `JSXSCAL` – "Jumpstart $\vec{\mathbf{x}}$ Scaling" keyword.

`JStart_X_Scal` – Type of state vector scaling to use before running the jumpstart solver at every time ministep (both in-solver and pre-solver scaling is considered below). It can be one of the following keywords:

- `-1` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use the jumpstart state vector scaling specified by the time integrator, if any. This is the default option.
- `0` – Do not use any jumpstart state vector scaling.
- `1` – Use pre-solver initial guess-based jumpstart state vector scaling.
- `2` – Use initial guess-based `TypicalX` solver option.
- `3` – Use Jacobian column sum-based jumpstart state vector scaling. This type of scaling will require the construction of the Jacobian matrix based on initial guess state vector. The jumpstarter may also use this Jacobian, if it is built, for additional solver options.

XXIII. `JSXCENT` – "Jumpstart $\vec{\mathbf{x}}$ Centering" keyword.

`JStart_X_Cent` – Type of state vector centering to use before running the jumpstart solver at every time ministep (the solver itself may implement additional centering operations, separate from these options; below only the pre-solver scaling is considered). It can be one of the following keywords:

- `-1` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use the jumpstart state vector centering specified by the time integrator, if any. This is the default option.
- `0` – Do not use any jumpstart state vector centering.
- `1` – Use initial guess-based jumpstart state vector centering, with centering about the zero vector.
- `2` – Use initial guess-based jumpstart state vector centering, with centering about the unity vector.

XXIV. `JSPC` – "Jumpstart Solver Preconditioner" keyword.

`JStart_PreCon` – Type of residual function preconditioner to use before running the jumpstart solver at every time ministep (the solver itself may implement additional preconditioning operations, separate from these options; below only the pre-solver preconditioning is considered). It can be one of the following keywords:

- `-1` – Either ministep-based jumpstart is not used or required by the specified time integrator, or use the jumpstart solver preconditioning specified by the time integrator, if any. This is the default option.
- `0` – Do not use any jumpstart solver preconditioning.
- `1` – Use Jacobian ILU jumpstart solver preconditioner. This preconditioner will require the construction of the Jacobian matrix based on initial guess state vector. The jumpstarter may also use this Jacobian, if it is built, for additional solver options.

XXV. `JSJACOB` – "Jumpstart Jacobian" keyword.

`JStart_Jacobian` – Whether, and how, an externally-constructed Jacobian is supplied to the jumpstart solver. It can be one of the following keywords:

- `-1` – Use the jumpstart solver Jacobian setting specified by the time integrator, if any. This is the default option.
- `0` – Do not supply an externally-constructed Jacobian to the jumpstart solver.
- `1` – Construct a Jacobian function based on MATLAB's built-in `numjac` function to pass to the jumpstart solver. State absolute tolerance is used as threshold values vector. No typical state vector is provided.

- `2` – Construct a Jacobian function based on MATLAB's built-in `numjac` function to pass to the jumpstart solver. State absolute tolerance is used as threshold values vector. Initial guess is provided as the typical state vector.

XXVI. `JSJSP` – "Jumpstart Solver Jacobian Sparsity" keyword.

`JStart_Jac_Sparsity` – Whether, and how, the Jacobian sparsity is treated during the jumpstart. It can be one of the following keywords:

- `-1` – Jacobian sparsity treatment during the jumpstart is specified by the time integrator, if any. This is the default option.
- `0` – Jumpstart Jacobian is treated as a full matrix.
- `1` – Jumpstart Jacobian is treated as a sparse matrix. The sparsity pattern is constructed exactly during sorting.
- `2` – Jumpstart Jacobian is treated as a sparse matrix. The sparsity pattern is obtained by using MATLAB's built-in `numjac` function on the initial state vector. This sparsity pattern is assumed to stay constant throughout the run. It is often incomplete, so option `1` is preferred.

Currently, the Jacobian sparsity pattern is always constructed exactly during sorting; it is supplied to the built-in MATLAB time integrators.

XXVII. `ADDJSO` – "Additional Jumpstart Solver Options" keyword.

`Addtnl_JSO_Num` – Number of additional jumpstart solver options pairs listed below. If `0`, no additional jumpstart solver options are listed below.

XXVIII. `ADDJSO_Pairs_List` – List of (parameter name, "parameter value") pairs, one pair per line, of additional jumpstart solver options. These parameters are normally the same as the ones specified in MATLAB's built-in `optimset` function arguments. The parameter values are listed in double quotes, and are executed (with `eval`) prior to being submitted as solver options, therefore they can contain references to variables or objects contained in Linked Files and Directories lists. If a parameter must stay a string after having `eval` executed on it, the parameter must have single quotation marks placed around it, in addition to the double quotation marks. This list is only present if `ADDJSO` is specified as greater than zero.

XXIX. `SJSOEND` – "Solver and Jumpstart Options End" keyword.

5. Evaluator Options

```
EO
CS2N If_CS2N
NFT NF_Type (Optional)
EOEND
```

Comments

I. `EO` – "Evaluator Options" keyword.

II. `CS2N` – "Convert Symbolic To Numeric" keyword.

`If_CS2N` – Whether or not to convert the symbolic expressions in the state derivative and bond variable vectors to MATLAB numeric expressions. These expressions are used in the time integrator and the post processor. `If_CS2N` can be one of the following keywords:

- `0` – Keep the expressions symbolic.

- `1` – Convert the expressions to numeric. This is the default option.

Only option `1` is currently supported!

III. `NFT` – "Numeric Function Type" keyword. (This line is only present with `CS2N` of `1`).
`NF_Type` – Type of MATLAB function to convert the symbolic expression to. `NF_Type` can be one of the following keywords:
- `0` – Convert the symbolic expressions into anonymous MATLAB function handles.
- `1` – Convert the symbolic expressions into temporary MATLAB .m function files. The function files are saved into the `<\TmpFunctions>` subdirectory of the directory the BGSD file is in. This is the default option.
- `2` – Convert the symbolic expressions into MATLAB .mex function files. The function files are saved into the `<\TmpFunctions>` subdirectory of the directory the BGSD file is in.

Only option `1` is currently supported!

IV. `EOEND` – "Evaluator Options End" keyword.

6. Post-Processor Options

`PPO`
`EPP Number_Efforts`
`Effort_List` (*Optional*)
`FPP Number_Flows`
`Flow_List` (*Optional*)
`PPDISP PP_Display` (*Optional*)
`PPOEND`

Comments

I. `PPO` – "Post-Processor Options" keyword.
II. `EPP` – "Efforts Post-Process" keyword.
`Number_Efforts` – Number of effort variables to evaluate during the post-processing. `Number_Efforts` can be one of the following keywords:
- `-1` – Evaluate all effort variables during the post-processing. This is the default option.
- `0` – Do not evaluate any effort variables during the post-processing.
- `number` – The positive number of effort variables to evaluate during the post-processing. The bond IDs of the effort variables to post-process are specified in the following line.

III. `Effort_List` – List of bond IDs of effort variables to evaluate during the post-processing, in order of increasing bond IDs, one entry per line. (This list is only present if a positive number is specified in `EPP`).
IV. `FPP` – "Flows Post-Process" keyword.
`Number_Flows` – Number of flow variables to evaluate during the post-processing. `Number_Flows` can be one of the following keywords:
- `-1` – Evaluate all flow variables during the post-processing. This is the default option.
- `0` – Do not evaluate any flow variables during the post-processing.
- `number` – The positive number of flow variables to evaluate during the post-processing. The bond IDs of the flow variables to post-process are specified in the following line.

V. `Flow_List` – List of bond IDs of flow variables to evaluate during the post-processing, in order of increasing bond IDs, one entry per line. (This list is only present if a positive number is specified in `FPP`).

VI. `PPDISP` – "Post-Processor Display" keyword. (This line can only be present if at least one post-processing variable is specified by either `EPP` or `FPP` options).
`PP_Display` – Post-processor level of display. Can be one of the following keywords:
- `0` – No time stepwise information is output during post-processing.
- `1` – Some statistical information is output after every time step during post-processing.

All outputs specified by this option are output only to the command window, regardless of the specified processor verbosity.

VII. `PPOEND` – "Post-Processor Options End" keyword.

7. Linked Files and Directories

```
LFD
LF Number_Files
Linked_Files_List (Optional)
LD Number_Directories
Linked_Directories_List (Optional)
LFDEND
```

Comments

I. `LFD` – "Linked Files and Directories" keyword.

II. `LF` – "Linked Files" keyword.
`Number_Files` – Number of linked files.

III. `Linked_Files_List` – List of .m and .mat files. This entry is only present if `LF` is greater than zero. The .m files are scripts which have to be executed prior to processing the BGSD file. These scripts are executed in the order they are linked in. It is convenient to use function handle declaration inside this script, because those function handles will then be accessible to the processor, even if the functions that the script links to are not in a linked directory. The .mat files contain data necessary for processing this BGSD file. The files are listed one file per line, in double quotes. Either full paths can be specified, or only the file names. If only the file names are specified, it is assumed that they are either in the directory the BGSD file is in, or in the `<\LinkedResources>` subdirectory of the directory the BGSD file is in. The directory the BGSD file is in is checked first.

IV. `LD` – "Linked Directories" keyword.
`Number_Directories` – Number of linked directories.

V. `Linked_Directories_List` – List of the directories which must be included in the MATLAB path to process this BGSD file, one directory per line, in double quotes. This entry is only present if `LD` is greater than zero. Either full paths can be specified, or only the directory names. If only the directory names are specified, it is assumed that they are in the directory the BGSD file is in, or in the `<\LinkedResources>` subdirectory of the directory the BGSD file is in. The directory that the BGSD file is in is checked first.

VI. `LFDEND` – "Linked Files and Directories End" keyword.

8. Additional System Information

```
ASI
SCI SC_Info
SMI SM_Info
SLI SL_Info
SRI SR_Info
SEI SE_Info
ASIEND
```

Comments

I. `ASI` – "Additional System Information" keyword.

II. `SCI` – "Specified Causality Information" keyword.
`SC_Info` – Specified information about causality of the BGS. This can be one of the following keywords:
- `FC` – "Full Causality." (Default option).
- `UC` – "Unknown Causality."

Processor v1.03 only works with BGSDs that are known to be fully causal, so any `SC_Info` other than `FC` will stop the processing and throw an error.

III. `SMI` – "Specified Modulation Information" keyword.
`SM_Info` – Specified information about the modulated elements of the BGS. This can be one of the following keywords:
- `NMEP` – "No Modulated Elements Present."
- `TMEP` – "Time-Modulated Elements Present."
- `SMEP` – "Signal-Modulated Elements Present."
- `UMP` – "Unspecified if Modulation Present." (Default option).

Processor v1.03 does not use this information, but it may be used in future versions.

IV. `SLI` – "Specified Linearity Information" keyword.
`SL_Info` – Specified information about linearity of the BGS. This can be one of the following keywords:
- `ACC` – "All Constant Coefficients" – All elements' expressions are constant coefficients.
- `ABSCON` – "All But Source Constant" – Source elements' expressions may be time-modulated, the other elements' expressions are constant coefficients.
- `NCC` – "Non-Constant Coefficients" – Expressions other than constant coefficients may be present.
- `UL` – "Unspecified Linearity." (Default option).

Processor v1.03 does not use this information, but it will be used in future versions.

V. `SRI` – "Specified Reduction Information" keyword.
`SR_Info` – Specified information about whether or not the BGS should be reduced. A reduction of a BGS refers to replacing it with an alternative BGS with identical state equations, but fewer bonds, elements, or both. Typically this is achieved by eliminating bypassed 1-port elements, or replacing a junction structure with a simpler equivalent junction structure. `SR_Info` can be one of the following keywords:
- `FR` – "Fully Reduced."
- `UR` – "Unknown if Reduced." (Default option).

Processor v1.03 does not use this information, but it will be used in future versions.

VI. `SEI` – "Specified Error Information" keyword.

`SE_Info` – Specified information about whether or not to check the BGS for errors. Examples of errors include 1-port elements with multiple bonds connected to them, 2-port elements with any number of bonds other then 2 connected to them, and similar errors that prevent the BGSD from making physical sense. `SE_Info` can be one of the following keywords:

- `CE` – "Check for Errors."
- `DNCE` – "Do Not Check for Errors." (Default option).

Processor v1.03 only works with BGSDs that are assumed to be error-free, so any `SE_Info` other than `DNCE` will stop the processing and throw an error.

VII. `ASIEND` – "Additional System Information End" keyword.

9. Bond Connectivity Map

```
BCM Number
Table
BCMEND
```

Comments

I. `BCM` – "Bond Connectivity Map" keyword.
`Number` – N_b, Number of bonds in the BGS.

II. `Table` – Table which stores the bond connectivity map as N_b lines of the following form, one line per bond, in order of increasing bond IDs:

```
To From
```

`To` – EID of the element the bond described on this line points To.

`From` – EID of the element the bond described on this line points From.

III. `BCMEND` – "Bond Connectivity Map End" keyword.

10. Element List

Notice, that the blank lines are not in the BGSD, and are given here just for simplifying reading. The indented lines are also only indented to simplify reading and are not indented in the BGSD.

```
EL Element_Number
EEXPRS Expr_Number
EMODS Mod_Number
EPAMBS Port_Ambig_Number
EBAMBS Bond_Ambig_Number
ECAMBS Caus_Ambig_Number
ENUMS Num_Elem_Number
NVARS Num_Var_Number
EV0S EV0_Number
ECAPS Num_Caps
EINERTS Num_Inerts
EPHXSPEC Phx_Spec_Number
ENNSPEC NN_Spec_Number
EATSPEC AbsTol_Spec_Number
```

`Element_Blocks`

`ELEND`

Comments

I. `EL` – "Element List" keyword.
`Element_Number` – Number of bond graph elements in the BGSD.

II. `EEXPRS` – "Elements with Expressions" keyword.
`EXPR_Number` – Number of elements with expressions in the BGSD. All elements except for 1- and 0-junctions have expressions.

III. `EMODS` – "Modulated Elements" keyword.
`Mod_Number` – Number of modulated elements in the BGSD.

IV. `EPAMBS` – "Elements with Ambiguous Port numbers" keyword.
`Port_Ambig_Number` – Number of elements with ambiguous port numbers in the BGSD. Presently, this includes only RN and MRN elements.

V. `EBAMBS` – "Elements with Ambiguous Bonds" keyword.
`Bond_Ambig_Number` – Number of elements on which it is ambiguous, which bonds are connected to which ports. Presently, this includes only RN and MRN elements. (R2 and MR2 elements have specific input-output convention; if the bond directionality on a 2-port resistive element cannot follow this convention, use an RN or MRN element instead, with $N = 2$).

VI. `ECAMBS` – "Elements with Ambiguous Causalities" keyword.
`Caus_Ambig_Number` – Number of elements with ambiguous causalities on their ports. Presently, this includes:
- R and MR elements with CE, NE, CME or NME expression types.
- R2, MR2, RN and MRN elements, with any expression types.

VII. `ENUMS` – "Numerical Elements" keyword.
`Num_Elem_Number` – Number of elements with numeric expression types. Presently, this includes NMC, NE and NME expression types.

VIII. `NVARS` – "Numeric Variables" keyword.
`Num_Var_Number` – Number of numeric variables, associated with individual numerical coefficients and expressions. Presently, this includes:
- N numeric variables per R2, MR2, RN and MRN element with NE or NME expression types, where N is the number of ports on the element.
- N^2 numeric variables per R2, MR2, RN and MRN element with NMC expression type.
- 1 numeric variable per any other element with NMC, NE or NME expression types.

IX. `EV0S` – "Elements with Values at 0" keyword.
`EV0_Number` – Number of storage elements with initial values.

X. `ECAPS` – "Capacitive Elements" keyword.
`Num_Caps` – Number of capacitive elements.

XI. `EINERTS` – "Inertial Elements" keyword.
`Num_Inerts` – Number of inertial elements.

XII. `EPHXSPEC` – "Physics-Specific Elements" keyword.

`Phx_Spec_Number` – Number of physics-specific elements. In v1.03, this only includes unmodulated storage elements.

XIII. `ENNSPEC` – "Elements with Non-Negativity Specified" keyword.

`NN_Spec_Number` – Number of elements with potentially enforceable non-negative state variables. In v1.03, this only includes storage elements.

XIV. `EATSPEC` – "Elements with Absolute Tolerances Specified" keyword.

`AbsTol_Spec_Number` – Number of elements with specified absolute tolerances on their state variables. In v1.03, this only includes storage elements.

`Element_Blocks` consists of a block of information for every element in the BGSD. The elements are listed in order of increasing EIDs. Every such block has the following form:

`EID ID_Number`
`ET Element_Type`
`EXPRT Expression_Type` (*If element has an expression*)
`V0 Initial_Value` (*If element is a storage element*)
`EP Port_Number` (*If port number is ambiguous*)
`EB Connected_BIDs` (*If port-to-bond connectivity is ambiguous*)
`EC Causality` (*If causality is ambiguous*)
`EPHX Physics_ID` (*If element is physics-specific*)
`ENN IfNonNeg` (*If element is has potentially non-negative state variable*)
`EAT AbsTol` (*If element has absolute tolerance on state variable*)
`EMVARS List` (*If element is modulated*)
`Expression(s)` (*If element has one or more expressions*)
`EEND`

XV. `EID` – "Element ID" keyword.

`ID_Number` – Unique identification number of the element whose expression follows. Elements are listed in order of increasing EIDs.

XVI. `ET` – "Element Type" keyword.

`Element_Type` – Element's type, can be one of the following 20 keywords:
`SE`, `SF`, `I`, `C`, `R`, `TF`, `GY`, `1`, `0`, `MSE`, `MSF`, `MI`, `MC`, `MR`, `MTF`, `MGY`, `R2`, `MR2`, `RN`, `MRN`

XVII. `EXPRT` – "Expression Type" keyword.

`Expression_Type` – Element's expression type, can be one of the following 7 keywords:
`CC`, `CMC`, `NMC`, `CE`, `NE`, `CME`, `NME`

XVIII. `V0` – "Value at 0" keyword.

`Initial_Value` – Element's initial value.

XIX. `EP` – "Element Ports" keyword.

`Port_Number` – Number of ports the element has.

XX. `EB` – "Element Bonds" keyword.

`Connected_BIDs` – List of bond IDs connected to the element's ports, in order of increasing port IDs.

XXI. `EC` – "Element Causality" keyword.

`Causality` – Element's causality vector. For 2-port elements the vector is in order of input, output. For *N*-port elements the vector is in order of increasing port IDs. Causality

vector consists of a single entry for every port of the element, with the following possible values:

- `1` – Effort is input on the port.
- `2` – Flow is input on the port.

XXII. `EPHX` – "Element Physics" keyword.

`Physics_ID` – Element physics ID. Physics ID is a single unsigned integer, with the following possible values:

- `0` – Unspecified physics ID. These elements are not assigned to a specific physics, and will be automatically assigned to the highest physics ID during processing. If all physics-specific elements have this element physics ID, this is identical to automatic full coupling, even if operator split is requested during integration.
- `Phx_ID` – A positive, specified physics ID.

XXIII. `ENN` – "Element Non-Negativity" keyword.

`IfNonNeg` – Element's state variable's enforced non-negativity. `IfNonNeg` can be one of the following possible values:

- `0` – Element's state variable is not necessarily non-negative. Note, that not all time integrators can enforce non-negativity.
- `1` – Element's state variable must be non-negative.

XXIV. `EAT` – "Element Absolute Tolerance" keyword.

`AbsTol` – Absolute tolerance on the element's state variable. If `0`, the time integrator's default absolute tolerance is used.

XXV. `EMVARS` – "Element Modulating Variables" keyword.

`List` – Double-quoted, comma-separated list of variables modulating the element. Time, if present, must be the first element in the list. They are in the following format:

- `t` – Time.
- `eN` – Effort on bond #N.
- `fN` – Flow on bond #N.
- `qN` – Displacement on capacitive element with EID N.
- `pN` – Momentum on inertial element with EID N.

XXVI. `Expression(s)` – Element's expressions. The expression format depends on the expression type and the element's type. The formats are presented below:

- For all elements other than R2, MR2, RN and MRN each element has one constituent expression.
    - For CC expression type:

`Coefficient`

    - For CMC expression type:

`Mod_CF_Expression`

    - For NMC expression type:

`Mod_Function_Handle`

    - For CE expression type:

`CF_Expression`

    - For NE expression type:

`Function_Handle`

    - For CME expression type:

`Mod_CF_Expression`

- For NME expression type:

`Mod_Function_Handle`

- For R2 and MR2 elements, each element has two or four constituent expressions.
  - For CC expression type, matrix of the following form:

`Coefficient1  Coefficient2`
`Coefficient3  Coefficient4`

  - For CMC expression type:

`Mod_CF_Expression1`
`Mod_CF_Expression2`
`Mod_CF_Expression3`
`Mod_CF_Expression4`

  - For NMC expression type:

`Mod_Function_Handle1`
`Mod_Function_Handle2`
`Mod_Function_Handle3`
`Mod_Function_Handle4`

  - For CE expression type:

`CF_Expression1`
`CF_Expression2`

  - For NE expression type:

`Function_Handle1`
`Function_Handle2`

  - For CME expression type:

`Mod_CF_Expression1`
`Mod_CF_Expression2`

  - For NME expression type:

`Mod_Function_Handle1`
`Mod_Function_Handle2`

- For RN and MRN elements, each element has either N^2 or N constituent expressions, depending on the element's expression type.
  - For CC expression type:

`Coefficient1_1` … `Coefficient1_N`
⋮
`CoefficientN_1` … `CoefficientN_N`

  - For CMC expression type:

`Mod_CF_Expression1`
⋮
`Mod_CF_ExpressionN`^{2}

  - For NMC expression type:

`Mod_Function_Handle1`
⋮
`Mod_Function_HandleN`^{2}

  - For CE expression type:

`CF_Expression1`

⋮

`CF_ExpressionN`

- For NE expression type:

`Function_Handle1`

⋮

`Function_HandleN`

- For CME expression type:

`Mod_CF_Expression1`

⋮

`Mod_CF_ExpressionN`

- For NME expression type:

`Mod_Function_Handle1`

⋮

`Mod_Function_HandleN`

The keywords are described below:

a. `Coefficient` – A single number.

b. `Mod_CF_Expression` – A MuPAD symbolic expression in double quotes for the expression being defined. It is a one-line expression which involves some of the modulating parameters listed in the `EMVARS` line, as well as (possibly) the independent variable(s). It does not involve the quantity actually being defined (the output), as that is assumed, based on the causality. So, if trying to define an expression of the format $f(x) = x^2$, the `Expression` will be:

`"x^2"`

The independent variables for 1-port elements are specified according to the following:

- `q` – Displacement associated with the port (Capacitive element).
- `p` – Momentum associated with the port (Inertial element).
- `e` – Effort input associated with the port (Resistive element in Conductance causality).
- `f` – Flow input associated with the port (Resistive element in Resistance causality).

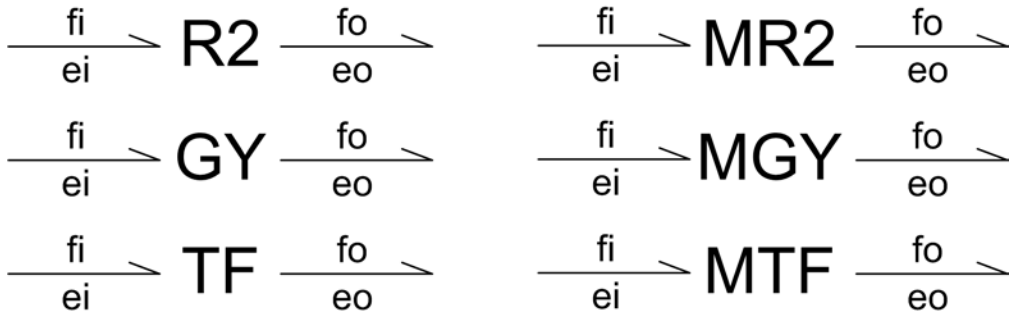

Figure 1. Bond Directionality Convention for 2-Port Elements

For 2-port elements, the `i` and `o` ("input" and "output," or "to" and "from") bonds are specified according to the convention in Figure 1 above. Notice that "input" bonds still have an output quantity, and "output" bonds still have an input quantity, according to the bonds' causalities. Here, "input" and "output" is just a naming convention for the bonds' directions.

- `ei` – Effort input on the input bond of a 2-port element.

- `fi` – Flow input on the input bond of a 2-port element.
- `eo` – Effort input on the output bond of a 2-port element.
- `fo` – Flow input on the output bond of a 2-port element.

For RN and MRN elements, the independent variables are as follows:

- `epj` – Effort on bond connected to port # j.
- `fpj` – Flow on bond connected to port # j.

For modulated elements, the following modulating independent variables are also used (all modulating variables have to be listed in the `EMVARS` line):

- `t` – Time.
- `eN` – Effort on bond # N.
- `fN` – Flow on bond # N.
- `qN` – Displacement on capacitive element with EID N.
- `pN` – Momentum on inertial element with EID N.

c. `Mod_Function_Handle` – A MATLAB function handle. The function handle's arguments are in the following order:

- Regular input variable(s), if the function belongs to an element with MNE expression type. For a 2-port element, the vector's elements are arranged as a 2×1 vector. For a 2-port element, the vector's elements are first that from the input bond, then that from the output bond. For an N-port element, the vector's elements are arranged as an $N\times1$ vector, with the variables in the vector's elements in the same order as in the `EB` list (that is, in order of increasing port IDs).
- Time, if the function is time-modulated.
- Modulating variables, in a vertical vector, if the function is modulated by any variables other than time. The order of elements in that vector is the same as the order of variables in the `EMVARS` line, excluding time.

`Mod_Function_Handle` must be vectorized with respect to regular input and modulating variables (or one of the two, if it does not depend on the other), as follows:

$$\left[F_1, F_2 \ldots F_N\right] = f_m\left(\left[\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2 \ldots \vec{\mathbf{y}}_N\right], t, \left[\vec{\mathbf{m}}_1, \vec{\mathbf{m}}_2 \ldots \vec{\mathbf{m}}_N\right]\right)$$

in which f_m is the modulated function, $\vec{\mathbf{y}}_i$ is a vector of regular inputs, t is time, $\vec{\mathbf{m}}_i$ is a vector of modulating inputs, and F_i is the value of the modulated function corresponding to inputs $\left(\vec{\mathbf{y}}_i, t, \vec{\mathbf{m}}_i\right)$. $\left[\vec{\mathbf{g}}_1 \ldots \vec{\mathbf{g}}_N\right]$ here constitutes a columnwise assembly of vertical vectors $\vec{\mathbf{g}}_1 \ldots \vec{\mathbf{g}}_N$ into a matrix.

d. `CF_Expression` – A MuPAD symbolic expression in double quotes for the expression being defined. It is a one-line expression, similar to the `Mod_CF_Expression`, defined above, but without the modulating variables. Everything else is the same.

e. `Function_Handle` – A MATLAB function handle, similar to the `Mod_Function_Handle`, defined above, but without the modulating inputs or time-dependence. Everything else is the same; the function is also expected to be vectorized, but only with respect to the regular input variable(s).

At the end of each element block is the `EEND` keyword.

XXVII. `EEND` –"Element End" keyword.

After all elements in the BGSD are listed comes the `ELEND` keyword.

XXVIII. `ELEND` – "Element List End" keyword.

11.File End

`BGSDEND`

Comments

I. `BGSDEND` – “Bond Graph System Descriptor End” keyword.