

Additional and Revised Subprograms of the JUPITER API

Revisions to the JUPITER API are made to be backward-compatible. That is, an application developed with an earlier version of the API should work correctly when compiled with a later version of the API, although formatting and other minor details of output may differ. This document is current with version 1.7.3 of the JUPITER API.

Dependents (DEP) Module

An array named STATISTIK was added at JUPITER version 1.4.0 and made public. This one-dimensional array holds the value of the variable STATISTIC for each used observation.

Subroutine DEP_GET_WTMULTIPLIER (added at JUPITER version 1.4.0)

Description: This subroutine returns the weight multiplier value (variable WTMULTIPLIER of the OBSERVATION_GROUPS input block) associated with a specified observation group.

Argument list:

(GPNAM, WTMULT)

Declarations for argument-list variables:

```
CHARACTER(LEN=12), INTENT(IN) :: GPNAM  
DOUBLE PRECISION, INTENT(OUT) :: WTMULT
```

Explanation of arguments:

GPNAM is the name of the observation group for which a weight multiplier value is required.
WTMULT is the weight multiplier for observation group GPNAM.

Equation (EQN) Module

Subroutine EQN_LINEAR_COEFFS (revised at JUPITER version 1.2.2)

Description: This subroutine parses a linear equation, and returns coefficients and constant terms.

It has been revised to optionally edit the prior information equation in the case where a log-transformed parameter appears in the equation but is not included in the parentheses of a log10 function. If CONVERT_STAT is present and true this routine will determine whether the equation is for a single parameter that is log transformed. If so and if the parameter does not appear as log10(parametername) the equation will be edited to be of that form. If it is already of that form, the code will proceed as usual. If such a parameter occurs in any other form, IFAIL is set to 1 and control is returned to the calling program unit.

Argument list:

(IFAIL, EQUATION_NAME, ATEXT, NVAR, NPE, AVAR, ITRANS, RVAR, CONST_TERM, WORK, XROW, ATEXT_TRANSFORM, CONVERT_STAT)

Declarations for argument-list variables:

INTEGER, INTENT(OUT)	:: IFAIL
CHARACTER (LEN=*), INTENT(IN)	:: EQUATION_NAME
CHARACTER (LEN=*), INTENT(IN)	:: ATEXT
INTEGER, INTENT(IN)	:: NVAR
INTEGER, INTENT(IN)	:: NPE
CHARACTER (LEN=*), INTENT(IN)	:: AVAR(NVAR)
INTEGER, INTENT(IN)	:: ITRANS(NVAR)
DOUBLE PRECISION, INTENT(IN)	:: RVAR(NVAR)
DOUBLE PRECISION, INTENT(OUT)	:: CONST_TERM
DOUBLE PRECISION, INTENT(OUT)	:: WORK(NVAR)
DOUBLE PRECISION, INTENT(OUT)	:: XROW(NPE)
CHARACTER (LEN=*), OPTIONAL, INTENT(OUT)	:: ATEXT_TRANSFORM
LOGICAL, OPTIONAL, INTENT(INOUT)	:: CONVERT_STAT

Explanation of arguments:

IFAIL signals an error condition. If it is returned as non-zero then an error condition has occurred. An error message will be available in the AMESSAGE string available from the Global Data Module. In this error message the equation will be identified by its name (12 characters or less) supplied by the calling program through the EQUATION_NAME variable.

EQUATION_NAME is the equation name.

ATEXT contains the text of the linear prior-information equation. It must contain only the part of the equation that can be evaluated; it must not contain an “=” sign.

NVAR is the number of parameters for which elements are populated in the AVAR, ITRANS and RVAR arrays.

NPE is the number of parameters that are adjustable through the parameter estimation process. This is also equal to the number of columns of the Jacobian matrix.

AVAR is an array of NVAR parameter names; corresponding parameter values need to be supplied in the RVAR array. Parameter names need to be lower case.

ITRANS contains the transformation state for each parameter in the AVAR array. An ITRANS value of 0 indicates that the pertinent parameter is untransformed, whereas a value of 1 indicates that it is log-transformed. A value of -10,000 (that is, negative 10,000) indicates that the parameter is fixed (at its corresponding RVAR value); any other negative value indicates that the parameter is tied, the index of the parent parameter being the negative of the value of ITRANS for that parameter. Exactly NPE elements of ITRANS must have a value of either 0 or 1.

RVAR contains the values of parameters identified in the AVAR array. Note that the only use subroutine EQN_LINEAR_COEFFS makes of the elements of the RVAR array is in the calculation of the constant term for the equation; fixed parameters contribute to that term.

CONST_TERM is the value taken by the linear prior-information equation when all adjustable parameters (or their logs) are assigned a value of zero. Thus it takes into account constant terms used in the equation as well as non-zero values of fixed parameters.

WORK is an array used by EQN_LINEAR_COEFFS only for internal processing.

XROW is populated with values that can be used to directly fill a row of the Jacobian matrix. It contains linear coefficients of adjustable parameters (that is, parameters for which ITRANS is either 0 or 1) in order of their appearance in the AVAR array. However, because the coefficients of only adjustable parameters are calculated, parameters whose ITRANS value is less than zero are omitted from this array. Elements pertaining to parameters not cited in a

linear prior-information equation are assigned a value of zero in the XROW array. It is important to note that where a parameter is log-transformed (that is, ITRANS for that parameter is supplied as 1), the pertinent element of the XROW array contains the coefficient of the log to base 10 of that parameter irrespective of whether the supplied equation uses the log of the parameter to base 10, or to base e .

ATEXT_TRANSFORM contains the text of the edited linear prior-information equation.

CONVERT_STAT is flag that indicates on entry whether the equation should be edited if a log-transformed parameter is found that is not in the parentheses of a log10 function, and indicates on exit whether such editing was required for the equation being evaluated.

Note: If ATEXT_TRANSFORM is present in argument list, CONVERT_STAT also must be present.

Prior_Information (PRI) Module

Coding was added to accommodate the case where additional parameters are considered for evaluation of predictive uncertainty. This is appropriate when parameters cannot be included in the regression because the simulated equivalents have little or no sensitivity to them, yet the predictions are sensitive to those parameters. In this case uncertainty associated with those parameters can be included by using the features added in this version of the API.

New Input Blocks

Two new Input blocks are read by the Prior-Information Module. Neither is required. Their position, if present, immediately follows the original block of the same name without the "_For_Prediction". The new blocks are as follows:

```
PRIOR_INFORMATION_GROUPS_FOR_PREDICTION ListPointer=LLPTRGPPRIORFP  
LINEAR_PRIOR_INFORMATION_FOR_PREDICTION ListPointer=LLPTRPRIORFP
```

Information for the new blocks is exactly the same as that for blocks

PRIOR_INFORMATION_GROUPS and LINEAR_PRIOR_INFORMATION as outlined in Tables 16-15 and 16-16 in TM6E1.

New Public Data

The ListPointers for new input blocks are new public data. Declarations for new public variables:

```
TYPE (LLIST), POINTER :: LLPTRGPPRIORFP
```

```
TYPE (LLIST), POINTER :: LLPTRPRIORFP
```

Explanation of data:

LLPTRGPPRIORFP points to an input-block data structure of information for prior-information groups included for prediction. Each list contains information for one prior-information group.

LLPTRPRIORFP points to an input-block data structure of prior-information data included for prediction. Each list contains information for one prior-information equation.

New Private Data

Declarations for new public variables:

```
INTEGER    :: MPRWOP = 0
INTEGER    :: NPRIGPSFP = 0
INTEGER    :: NPRIGPSWOP = 0
```

Explanation of data:

MPRWOP is the number prior information items that were considered for a previous task (generally parameter estimation) .

NPRIGPSFP is the number prior information groups defined for an advanced task, for example for predictive evaluation following a parameter estimation task.

NPRIGPSWOP is the number prior information groups defined for a previous task, for example a parameter estimation preceding a predictive evaluation.

Subroutine PRI_INI_POPX (optional argument added at JUPITER version 1.5.0)

Description: This subroutine populates the sensitivity matrix for prior-information equations. If

IVERB of the Global Data Module is greater than 3, the prior-information sensitivity matrix and the prior-information weight matrix are written to unit IOUT. If the optional argument CONVERTED is present, it needs to contain an element for each prior-information equation. An element needs to be .TRUE. if the corresponding prior-information equation was entered by the user for a parameter that is to be log transformed for analysis but the equation was not of the form log10(paramname). In this case, it is assumed that the user has entered the statistic for the prior information in native space and that the prior-information equation has been edited by PRI_INI_STORE_WPRED (as documented below) to be of the form log10(paramname) before PRI_INI_POPX is invoked. PRI_INI_POPX will convert the statistic first to log base 10 and then to natural log space, because internal calculations are carried out in natural log space. If CONVERTED is .FALSE. and the parameter is not to be log-transformed, no conversions are made. If CONVERTED is .FALSE. and the parameter is to be log-transformed, it is assumed the user has entered the statistic in log10 space, and the statistic will be converted to natural log space. NOTE: If a full weight matrix is utilized by the user, it is assumed that it is input in a manner consistent with the parameter transform specifications. Elements of a full weight matrix are not converted.

Argument list:

(IOUT, IPRC, MPR, NPE, NPT, IPTR, LN, PARNAM, XLOG, XPRI, XPRI FILLED, CONVERTED)

Declarations for argument-list variables:

```
INTEGER,          INTENT( IN)    :: IOUT
INTEGER,          INTENT( IN)    :: IPRC
INTEGER,          INTENT( IN)    :: MPR
INTEGER,          INTENT( IN)    :: NPE
INTEGER,          INTENT( IN)    :: NPT
INTEGER,          INTENT( IN)    :: IPTR
INTEGER,          DIMENSION(NPE), INTENT( IN) :: IPTR
```

INTEGER,	DIMENSION(NPT),	INTENT(IN)	:: LN
CHARACTER(LEN=12),	DIMENSION(NPT),	INTENT(IN)	:: PARNAM
DOUBLE PRECISION,		INTENT(IN)	:: XLOG
DOUBLE PRECISION,	DIMENSION(NPE,MPR),	INTENT(INOUT)	:: XPRI
LOGICAL,		INTENT(INOUT)	:: XPRI FILLED
LOGICAL, OPTIONAL,	DIMENSION(MPR),	INTENT(IN)	:: CONVERTED

Explanation of arguments:

IOUT is the unit number associated with the file to which output is to be written.

IPRC is a code that defines the format to be used when the prior-information sensitivity matrix is written. Valid values of IPRC and corresponding output formats are documented in the description of UTL_WRITEMATRIX in the “Data Output” section of Appendix D.

MPR is the number of prior-information equations.

NPE is the number of estimated parameters.

NPT is the total number of parameters.

IPTR is the position of parameter in the full set of parameters.

LN is a flag indicating whether the parameter is log-transformed.

PARNAM contains the parameter names.

XLOG sensitivity in log space for transformed parameters.

XPRI sensitivity array for prior information.

CONVERTED (if present) contains an element for each prior-information equation. An element needs to be .TRUE. if corresponding prior-information equation has been edited to be of the form $\log_{10}(\text{paramname})$.

Subroutine PRI_INI_READ_WPRED (added at JUPITER version 1.2.2)

Description: This subroutine can be used in place of PRI_INI_READ; it calls UTL_READBLOCK to read four input blocks. The first block is labeled “Prior_Information_Groups”; the REQUIRED argument of UTL_READBLOCK is false, so the “Prior_Information_Groups” input block is optional. If present, the “Prior_Information_Groups” input block is expected to contain information related to groups of prior-information equations. The keyitem for this input block is “GROUPNAME”. The second block is the “Prior_Information_Groups_For_Prediction” and also is optional. If present, the “Prior_Information_Groups_For_Prediction” input block is expected to contain information related to groups of prior-information equations that are used for predictive analysis and not for parameter estimation. The keyitem for this input block is “GROUPNAME”. The third block is the “Linear_Prior_Information” block and also is optional. The keyitem for this input block is “PRIORNAME”. If present, it is expected to contain information needed to define individual prior-information equations. The fourth block is the “Linear_Prior_Information_For_Prediction” block and also is optional. The keyitem for this input block is “PRIORNAME”. If present, it is expected to contain information needed to define individual prior-information equations that are used for predictive analysis and not for parameter estimation. If both of the respective companion (Groups and Prior Information) input blocks are present, the keywords and corresponding values in the Groups block are inserted into the input-block data structure holding data from the Prior_Information block according to matching GROUPNAME values. Group blocks do not have a default column

order and the new “Linear_Prior_Information_For_Prediction” block has the same default order as the “Linear_Prior_Information” block.

Argument list:

(INUNIT, IOUT, IWRITE, NPE, NCOVMAT, MPR, PRED, MPRWOPPAS)

Declarations for argument-list variables:

```
INTEGER, INTENT(IN)      :: INUNIT
INTEGER, INTENT(IN)      :: IOUT
LOGICAL, INTENT(IN)      :: IWRITE
INTEGER, INTENT(IN)      :: NPE
INTEGER, INTENT(INOUT)   :: NCOVMAT
INTEGER, INTENT(OUT)     :: MPR
LOGICAL, INTENT(IN)      :: PRED
INTEGER, INTENT(OUT)     :: MPRWOPPAS
```

Explanation of arguments:

INUNIT is the unit number associated with the file from which input is to be read.

IOUT is the unit number associated with the file to which output is to be written.

IWRITE is a flag indicating whether the number of prior-information equations should be written to IOUT.

NPE is the number of adjustable parameters.

NCOVMAT is the number of variance-covariance matrices to be read.

MPR is the number of prior-information equations read from the “Linear_Prior_Information” input block.

PRED, when specified as .TRUE., indicates that the PRIOR_INFORMATION_GROUPS_FOR_PREDICTION and LINEAR_PRIOR_INFORMATION_FOR_PREDICTION input blocks, if present, should be read and that module arrays (PRIGROUPNAM, PRIUSEFLAG, PLOTSYMBOLPRIGRP, PRIWTMULTIPLIER, PRINAM, PRIVAL, PRIVARIANCE, PRIGROUP, PLOTSYMBOLPRI, PRICOVMATNAM) should be allocated to accommodate the additional prior-information data.

MPRWOPPAS is the number prior-information items that are defined in the LINEAR_PRIOR_INFORMATION block.

Subroutine PRI_INI_STORE_WPRED (subroutine added at JUPITER version 1.2.2;

optional argument added at version 1.5.0)

Description: This subroutine stores the prior-information data in arrays and can be used in place of PRI_INI_STORE. The revision accommodates two features not supported by PRI_INI_STORE. First, a logical flag indicates whether a prior-information equation should be modified if the parameter involved is log transformed and the parameter is not included in parentheses after a log10 function in the prior equation. In addition, the subroutine accommodates new prior information may be present for the predictive mode. If the optional argument CONVERTED is included then it will defined as .TRUE. if the subroutine EQN_LINEAR_COEFFS edits the prior information equation from the form (paramname) to

the form $\log_{10}(\text{paramname})$. It will be defined as `.FALSE.` if `EQN_LINEAR_COEFFS` does not edit the equation.

Argument list:

(`IOUT`, `MPR`, `NPE`, `NPT`, `ITRANS`, `PARNAMLC`, `PVAL`, `XPRIFILLED`, `CONVERT_STAT`, `PRED`, `MPRWOPPAS`, `CONVERTED`)

Declarations for argument-list variables:

<code>INTEGER,</code>	<code>INTENT(IN)</code>	<code>:: IOUT</code>
<code>INTEGER,</code>	<code>INTENT(IN)</code>	<code>:: MPR</code>
<code>INTEGER,</code>	<code>INTENT(IN)</code>	<code>:: NPE</code>
<code>INTEGER,</code>	<code>INTENT(IN)</code>	<code>:: NPT</code>
<code>INTEGER, DIMENSION(NPT),</code>	<code>INTENT(IN)</code>	<code>:: ITRANS</code>
<code>CHARACTER(LEN=12), DIMENSION(NPT),</code>	<code>INTENT(IN)</code>	<code>:: PARNAMLC</code>
<code>DOUBLE PRECISION, DIMENSION(NPT),</code>	<code>INTENT(IN)</code>	<code>:: PVAL</code>
<code>LOGICAL,</code>	<code>INTENT(INOUT)</code>	<code>:: XPRIFILLED</code>
<code>LOGICAL,</code>	<code>INTENT(INOUT)</code>	<code>:: CONVERT_STAT</code>
<code>LOGICAL,</code>	<code>INTENT(IN)</code>	<code>:: PRED</code>
<code>INTEGER,</code>	<code>INTENT(IN)</code>	<code>:: MPRWOPPAS</code>
<code>LOGICAL, OPTIONAL, DIMENSION(MPR),</code>	<code>INTENT(INOUT)</code>	<code>:: CONVERTED</code>

Explanation of arguments:

`IOUT` is the unit number associated with the file to which output is to be written.

`MPR` is the number of prior-information items.

`NPE` is the number of adjustable parameters.

`NPT` is the number of parameters.

`ITRANS` contains the transformation flag for each parameter, as required for subroutine `EQN_LINEAR_COEFFS` of the Equation Module.

`PARNAMLC` contains the parameter names, in all lowercase.

`PVAL` contains the current parameter values.

`XPRIFILLED` is a flag indicating whether the linear prior-information sensitivity array has been filled as it need only be done once.

`CONVERT_STAT` is a logical flag indicating whether a prior information equation should be modified if the parameter involved is log transformed and the parameter is not included in parentheses after a \log_{10} function in the prior equation. If such correction is necessary the equation is revised and the revised equation will be written to the “_pr” file. If its value returns as “true” then the variance and weight of the prior information should be converted to log space. This can be accomplished with the revised version of `UTL_CALCWT`.

`PRED`, when specified as `.TRUE.`, indicates that new prior information may be present for the predictive mode.

`MPRWOPPAS` is the of number prior-information items that are defined in the `LINEAR_PRIOR_INFORMATION` block.

`CONVERTED` (if present) contains an element for each prior-information equation. An element will be assigned as `.TRUE.` if the corresponding prior-information equation has been edited to be of the form $\log_{10}(\text{paramname})$ and `.FALSE.` if it has not been edited.

Subroutine PRI_UEV_DX_READ_PRP (added at JUPITER version 1.2.2)

Description: This subroutine reads a data-exchange file of type “_prp” which contains prior information equations related to prior added after parameter estimation and considered only for predictive evaluations.

Argument list:

(MPRFP, OUTNAM, PLOTSYM, PRIEQNTXT, PRINAME, PRIVAL)

Declarations for argument-list variables:

INTEGER,	INTENT (IN)	:: MPRFP
CHARACTER (LEN=MAX_STRING_LEN),	INTENT (IN)	:: OUTNAM
INTEGER, DIMENSION (MPRFP),	INTENT (OUT)	:: PLOTSYM
CHARACTER (LEN=MAX_STRING_LEN), DIMENSION (MPRFP),	INTENT (OUT)	:: PRIEQNTXT
CHARACTER (LEN=LENDNAM), DIMENSION (MPRFP),	INTENT (OUT)	:: PRINAME
DOUBLE PRECISION, DIMENSION (MPRFP),	INTENT (OUT)	:: PRIVAL

Explanation of arguments:

MPRFP is number prior information items added for an advanced evaluation (generally predictive analysis) and read from the _prp file.

OUTNAM is the root name of the data-exchange file. .

PLOTSYM is populated with the plot-symbol values.

PRIEQNTXT is populated with the prior-information equations.

PRINAM is populated with the prior-information equation names.

PRIVAL is populated with the prior-information equation values.

Subroutine PRI_UEV_DX_WRITE_PRP (added at JUPITER version 1.2.2)

Description: This subroutine writes a data-exchange file of type “_prp” which contains prior information equations related to prior added after parameter estimation and considered only for predictive evaluations.

Argument list:

(MPR, MPRWOPPAS, OUTNAM)

Declarations for argument-list variables:

INTEGER,	INTENT (IN)	:: MPR
INTEGER,	INTENT (IN)	:: MPRWOPPAS
CHARACTER (LEN=*),	INTENT (IN)	:: OUTNAM

Explanation of arguments:

MPR is total number prior information items including those from the previous evaluation (generally parameter estimation) and those from the later evaluation (generally predictive analysis).

MPRWOPPAS is the number prior information items that were considered for the previous task (generally parameter estimation) .

OUTNAM is the root name of the data-exchange file.

Notes:

1. The number of items to be written to the _prp file is MPR-MPRWOPPAS.

Sensitivity (SEN) Module

Subroutine SEN_UEV_DX_READ_SU (revised at JUPITER versions 1.2.2 and 1.7.3)

Description: This subroutine reads unscaled sensitivities of dependents with respect to parameters from a data-exchange file of type _su, _supri, _suprip, _su_presvd, or _supri_presvd.

Argument list:

(NDEP, NPE, OBSNAM, OUTNAM, PARNAM, PLOTSYMBOL, XSSENS, EXT)

Declarations for argument-list variables:

INTEGER,		INTENT (IN)	:: NDEP
INTEGER,		INTENT (IN)	:: NPE
CHARACTER (LEN=MAX_STRING_LEN),		INTENT (IN)	:: OUTNAM
CHARACTER (LEN=LENDNAM),	DIMENSION (NDEP),	INTENT (OUT)	:: OBSNAM
CHARACTER (LEN=12),	DIMENSION (NPE),	INTENT (OUT)	:: PARNAM
INTEGER,	DIMENSION (NDEP),	INTENT (OUT)	:: PLOTSYMBOL
DOUBLE PRECISION,	DIMENSION (NPE, NDEP),	INTENT (OUT)	:: XSSENS
CHARACTER (LEN=*),	OPTIONAL,	INTENT (IN)	:: EXT

Explanation of arguments:

NDEP is the number of dependents.

NPE is the number of parameters.

OUTNAM is the root name for the data-exchange file.

OBSNAM is populated with the dependent names read from the data-exchange file.

PARNAM is populated with the parameter names read from the data-exchange file.

PLOTSYMBOL is populated with the plot-symbol values read from the data-exchange file.

XSSENS is populated with the unscaled sensitivities read from the data-exchange file.

EXT is the extension _su, _supri, _suprip, _su_presvd, or _supri_presvd.

Notes:

1. If EXT is not specified the default extension for the data-exchange file is _su. Otherwise, the data-exchange file name is formed by concatenating OUTNAM, '.', and EXT.
2. The files _supri and _suprip have the same format as _su. The file _supri contains the unscaled sensitivity for period defined in the LINEAR_PRIOR_INFORMATION block, while _suprip contains the unscaled sensitivity for prior contained in the LINEAR_PRIOR_INFORMATION_FORPREDICTION block. The latter contains additional columns equal to the number of additional prior-information items considered for prediction.
3. The files _su_presvd and _supri_presvd have the same format as _su. The file _su_presvd contains a copy of _su based on parameters used to start singular value decomposition (SVD), while _supri_presvd contains a copy of _supri based on parameters used to start SVD.

Utilities (UTL) Module

Function UTL_ACHAR (added at JUPITER version 1.3.0)

Description: Return 1-byte character represented by ASCII code I.

Argument list:

(I)

Result:

(A)

Declarations for argument-list and result variables:

```
INTEGER, INTENT( IN ) :: I
CHARACTER( LEN=1 )    :: A
```

Explanation of argument and result:

I is an ASCII code for a character.

A is the character represented by ASCII code I.

Note:

This function is needed because, for some compilers, the ACHAR function does not correctly handle a hard-coded reference like "ACHAR(2)"

Subroutine UTL_CALCWT (revised at JUPITER version 1.2.2)

Description: Calculate WEIGHT and VARIance associated with an observed or reference VALUE, using a STATISTIC. STATFLAG must be one of: 'CV', 'SD', 'VAR', 'WT', or 'SQRWT'. This modification includes an option argument, which if present and true will convert the variance from a native value to a log₁₀ value.

Argument list:

(IOU, NAME, STATFLAG, STATISTIC, VALUE, WTMULT, IERR, WEIGHT, VAR, CONVERT_STAT)

Declarations for argument-list and result variables:

```
INTEGER,          INTENT( IN )    :: IOU
CHARACTER( LEN=LENDNAM ), INTENT( IN ) :: NAME
CHARACTER( LEN=6 ), INTENT( IN )  :: STATFLAG
DOUBLE PRECISION, INTENT( IN )    :: STATISTIC
DOUBLE PRECISION, INTENT( IN )    :: VALUE
DOUBLE PRECISION, INTENT( IN )    :: WTMULT
INTEGER,          INTENT( INOUT ) :: IERR
DOUBLE PRECISION, INTENT( OUT )   :: WEIGHT
DOUBLE PRECISION, INTENT( OUT )   :: VAR
LOGICAL, OPTIONAL, INTENT( IN )   :: CONVERT_STAT
```

Explanation of arguments:

IOU is the unit number associated with the file to which error messages are to be written, if required.

NAME is the name associated with the observed or reference value.

STATFLAG indicates the type of statistic provided in the STATISTIC argument. The options are:

“CV” – Coefficient of variation

“SD” – Standard deviation

“VAR” – Variance

“WT” – Weight

“SQRWT” – Square root of the weight

STATISTIC is the value from which the weight is to be calculated.

VALUE is the observed or reference value to which the weight is to apply.

WTMULT is a value that will be used to multiply the weight.

IERR is incremented by 1 if an error is encountered in the subroutine.

WEIGHT is the weight.

VAR is the variance.

CONVERT_STAT should be specified as .TRUE. to indicate that the variance should be converted from native space to \log_{10} space.

Notes:

1. The formula used to calculate the weight is determined by the value of STATFLAG, as listed in the original API documentation.
2. Variance is calculated as the inverse of the weight.
3. Conversion from native variance to variance in \log_{10} space is as follows:

$$\sigma_{\log_{10} b}^2 = \frac{\ln \left[\left(\frac{\sigma_b}{b} \right)^2 + 1 \right]}{(\ln(10))^2}$$

Subroutine UTL_COPYLIST (added at JUPITER version 1.7.0)

Description: This subroutine makes a copy of data contained in a linked list.

Argument list:

(HEAD, COPY)

Declarations for argument-list variables:

TYPE (LLIST), POINTER :: HEAD

TYPE (LLIST), POINTER :: COPY

Explanation of arguments:

HEAD points to a linked list containing data to be copied.

COPY, on return, will point to a linked list containing a copy of data in the HEAD linked list.

Subroutine UTL_DIAGONAL_SUB (added at JUPITER version 1.1.0)

Description: This subroutine populates an array with the diagonal elements extracted from a square matrix stored in a compressed matrix structure.

Argument list:

(CDM, ARR)

Declarations for argument-list and result variables:

```
TYPE (CDMATRIX),                                INTENT(IN)    :: CDM
DOUBLE PRECISION, DIMENSION(CDM%NR), INTENT(OUT) :: ARR
```

Explanation of arguments:

CDM contains a square matrix.

ARR is populated with diagonal elements in CDM.

Subroutine UTL_DX_READ_MCMV (revised at JUPITER version 1.7.3)

Description: This subroutine reads a data-exchange file of type “_mc” or “_mv”. If the file to be read has been prepared correctly, the subroutine reads either a parameter correlation (_mc) or variance-covariance (_mv) matrix.

Argument list:

(EXT, NPE, OUTNAM, PARNAM, CMAT)

Declarations for argument-list variables:

```
CHARACTER(LEN=*) ,                                INTENT(IN)    :: EXT
INTEGER ,                                              INTENT(IN)    :: NPE
CHARACTER(LEN=*) ,                                INTENT(IN)    :: OUTNAM
CHARACTER(LEN=12) , DIMENSION(NPE) ,              INTENT(OUT)   :: PARNAM
DOUBLE PRECISION, DIMENSION(NPE,NPE) , INTENT(OUT) :: CMAT
```

Explanation of arguments:

EXT is the extension associated with the file type to be read. It must be either “_mc”, “_mv”, “_mc_presvd”, or “_mv_presvd”.

NPE is the number of adjustable parameters.

OUTNAM is the file-name base to be used in constructing the name of the data-exchange file to be read.

PARNAM is populated with parameter names read from the data-exchange file.

CMAT is populated with the matrix read from the data-exchange file. If EXT is “_mc”, CMAT is populated with a parameter correlation matrix. If EXT is “_mv”, CMAT is populated with a variance-covariance matrix. If EXT is “_mc_presvd”, CMAT is populated with a copy of a parameter correlation matrix based on parameters used to start singular value decomposition (SVD). If EXT is “_mv_presvd”, CMAT is populated with a copy of a variance-covariance matrix based on parameters used to start SVD.

Subroutine UTL_DX_READ_WT (revised at JUPITER version 1.7.3)

Description: This subroutine reads a data-exchange “_wt” or “_wtpri” file (Table 17-3) and populates one or two CDMATRIX structures (Chapter 1) with data read from the file. As of version 1.7.3, this subroutine also supports files of type “_wt_presvd” and “_wtpri_presvd”, which use the same format.

Argument list:

(IOUT, OUTNAM, EXT, WTMAT, WTMATSQR)

Declarations for argument-list and result variables:

```
INTEGER ,                                INTENT(IN)    :: IOUT
CHARACTER(LEN=*) ,                      INTENT(IN)    :: OUTNAM
```

```

CHARACTER ( LEN=* ) ,           INTENT ( IN )      :: EXT
TYPE ( CDMATRIX ) ,             INTENT ( INOUT )   :: WTMAT
TYPE ( CDMATRIX ) , OPTIONAL , INTENT ( INOUT )   :: WTMATSQR

```

Explanation of arguments:

IOUT is the unit number associated with the file to which error messages are to be written, if required.

OUTNAM is the root file name for output files.

EXT is one of two supported data-exchange file types to be created; its value may be either “_wt” or “_wtpri.” If EXT is “_wt”, matrix(ces) for dependents is (are) read from a data-exchange file of type _wt. If EXT is “_wtpri,” matrix(ces) for prior information is (are) read from a data-exchange file of type _wtpri (Table 17-3). With version 1.7.3, support is added for extensions “_wt_presvd” and “_wtpri_presvd,” which are for matrices of weights and prior information (respectively) for parameters used to start singular value decomposition.

WTMAT is populated with the first matrix in the input file with the base name OUTNAM.

WTMATSQR is populated with the second matrix in the input file with the base name OUTNAM, if WTMATSQR is present in the argument list and if the file contains two matrices.

Subroutine UTL_EIGEN (as revised at JUPITER version 1.6.0)

Description: Calculate eigenvalues and eigenvectors of a square matrix. Write an eigenvalue for each of NPE parameters and an NPExNPE table of eigenvectors to output.

Argument list:

```

(IOUT, IPRC, NPE, NPS, IPTR, ITERP, PARNAM, PVALINIT, PVAL, C, DATAEXCHANGE,
OUTNAM, PRINTIOUTIN)

```

Declarations for argument-list variables:

```

INTEGER ,                                INTENT ( IN )      :: IOUT
INTEGER ,                                INTENT ( IN )      :: IPRC
INTEGER ,                                INTENT ( IN )      :: NPE
INTEGER ,                                INTENT ( IN )      :: NPS
INTEGER ,          DIMENSION ( NPE ) ,    INTENT ( IN )      :: IPTR
INTEGER ,                                INTENT ( IN )      :: ITERP
CHARACTER ( LEN=12 ) , DIMENSION ( NPS ) , INTENT ( IN )      :: PARNAM
DOUBLE PRECISION , DIMENSION ( NPS ) ,    INTENT ( IN )      :: PVALINIT
DOUBLE PRECISION , DIMENSION ( NPS ) ,    INTENT ( IN )      :: PVAL
DOUBLE PRECISION , DIMENSION ( NPE , NPE ) , INTENT ( INOUT ) :: C
LOGICAL ,          OPTIONAL ,             INTENT ( IN )      :: DATAEXCHANGE
CHARACTER ( LEN=* ) , OPTIONAL ,           INTENT ( IN )      :: OUTNAM
LOGICAL ,          OPTIONAL ,             INTENT ( IN )      :: PRINTIOUTIN

```

Explanation of arguments:

IOUT is the unit number of the output file where the eigenvalues and eigenvectors will be written.

IPRC is a code that defines the format to be used when is written. Valid values of IPRC and corresponding output formats are documented in the description of UTL_WRITEMATRIX, below.

NPE is the number of adjustable parameters.

NPS is the total number of parameters, including adjustable and non-adjustable, primary or derived parameters.

IPTR contains, for each adjustable parameter, the position (element number) in the PARNAM, PVALINIT, and PVAL arrays corresponding to the adjustable parameter.

ITERP is the current iteration number.

PARNAM contains the parameter names.

PVALINIT contains the initial parameter values.

PVAL contains the current parameter values.

C is the variance/covariance matrix of the parameters.

DATAEXCHANGE indicates if an _eig data exchange file should be written. If true, the _eig file is written.

OUTNAM is the root name for the _eig data exchange file.

PRINTIOUTIN indicates the amount of information that should be written to IOUT. If true, the following items are printed: 1) Variance-Covariance matrix scaled with parameters, 2) Eigenvalues; and 3) Eigenvectors. If false, these items are not printed. If PRINTIOUTIN is not present, it defaults to true.

Note:

In this routine, C is scaled with the parameters, then the eigenvectors and eigenvalues are calculated.

Subroutine UTL_INVERT_DIAG (added at JUPITER version 1.1.0)

Description: This subroutine inverts a compressed, diagonal matrix.

Argument list:

(IFAIL, A, AS, DTLA)

Declarations for argument-list variables:

```
INTEGER,          INTENT(INOUT) :: IFAIL
TYPE (CDMATRIX), INTENT(INOUT) :: A
TYPE (CDMATRIX), INTENT(INOUT) :: AS
DOUBLE PRECISION, INTENT(OUT)   :: DTLA
```

Explanation of arguments:

IFAIL is a flag for error messaging, 0 indicates no problem, 1 indicates failure.

A on invocation contains the matrix to be inverted; on return, A contains the inverted matrix.

AS is populated with the square root of the inverse matrix.

DTLA is the log-determinant of the matrix.

Notes:

1. On invocation, A must contain no non-zero off-diagonal terms, and all terms on the diagonal must be greater than zero.
2. UTL_INVERT_DIAG is similar in function to UTL_SVD but is substantially more efficient than UTL_SVD. The difference is especially noticeable when the matrix dimension is large.

Subroutine UTL_MATMUL_SUB (added at JUPITER version 1.1.0)

Description: This generic subroutine interface performs matrix multiplication of an ordinary 2-D array and a compressed matrix.

Argument list:

(NR, NC, A, B, C)

Declarations for argument-list variables:

```
INTEGER,                                INTENT( IN)   :: NR
INTEGER,                                INTENT( IN)   :: NC
[various types],                        INTENT( IN)   :: A
[various types],                        INTENT( IN)   :: B
DOUBLE PRECISION, DIMENSION(:,:), INTENT( OUT) :: C
```

Explanation of arguments:

NR is the number of rows in the ordinary 2-D array, which may be either A or B.

NC is the number of columns in the ordinary 2-D array, which may be either A or B.

A is either an ordinary 2-D array of type DOUBLE PRECISION, with dimensions (NR,NC) or a compressed-matrix structure of type CDMATRIX (see Notes).

B is either an ordinary 2-D array of type DOUBLE PRECISION, with dimensions (NR,NC) or a compressed-matrix structure of type CDMATRIX (see Notes).

C is an ordinary 2-D array, with dimensions appropriate for the matrix product [A] x [B].

Notes:

If A is an ordinary array, then B must be a compressed-matrix structure, and the dimensions of C must be (NR,B%NC).

If A is a compressed-matrix structure, then B must be an ordinary 2-D array, and the dimensions of C must be (A%NR,NC).

Subroutine UTL_MATMULVEC_SUB (added at JUPITER version 1.1.0)

Description: This subroutine performs matrix multiplication of a compressed matrix times an ordinary 1-D array, where the 1-D array is assumed to represent a column vector.

Argument list:

(NRB, CA, OB, OC)

Declarations for argument-list variables:

```
INTEGER,                                INTENT( IN)   :: NRB
TYPE (CDMATRIX),                        INTENT( IN)   :: CA
DOUBLE PRECISION, DIMENSION(NRB),      INTENT( IN)   :: OB
DOUBLE PRECISION, DIMENSION(CA%NR),    INTENT( OUT)  :: OC
```

Explanation of arguments:

NRB is the dimension of the OB array; it must equal the NC component of the CA structure.

CA is the compressed-matrix structure to be multiplied by OB.

OB contains the column vector used to multiply CA.

OC is assigned as the matrix product [CA] x [OB].

Subroutine UTL_SLEEP (added at JUPITER version 1.7.2)

Description: This subroutine causes program execution to be suspended for a period of time (in seconds) provided as an argument.

Argument list:

(SECS)

Declaration for argument-list variable:

```
DOUBLE PRECISION, INTENT(IN) :: SECS
```

Explanation of arguments:

SECS is the time, in seconds, for which program execution is to be suspended.

Subroutine UTL_SUBSTITUTE_SUB (added at JUPITER version 1.1.0)

Description: This subroutine populates an array of dimension NPT with parameter values using current values in the PVAL array, but with substitution of values for all adjustable parameters provided in the PARVALSET array.

Argument list:

(NPE, NPT, IPTR, PARVALSET, PVAL, PVTMP)

Declarations for argument-list variables:

```
INTEGER,                                INTENT(IN)  :: NPE
INTEGER,                                INTENT(IN)  :: NPT
INTEGER, DIMENSION(NPE),                INTENT(IN)  :: IPTR
DOUBLE PRECISION, DIMENSION(NPE),        INTENT(IN)  :: PARVALSET
DOUBLE PRECISION, DIMENSION(NPT),        INTENT(IN)  :: PVAL
DOUBLE PRECISION, DIMENSION(NPT),        INTENT(OUT) :: PVTMP
```

Explanation of arguments:

NPE is the number of adjustable parameters.

NPT is the number of parameters.

IPTR contains, for each adjustable parameter, the position (element number) in the PVAL array corresponding to the adjustable parameter.

PARVALSET contains values of adjustable parameters, which may differ from the current parameter values, for example, for perturbation of a parameter.

PVAL contains the current (unperturbed) parameter values.

PVTMP is populated with values from PVAL with substitution of values in the PARVALSET array for all adjustable parameters.

Function UTL_SYSTEM_FUNC (added at JUPITER version 1.5.0)

Description: This logical function invokes an operating-system command. UTL_SYSTEM_FUNC uses the SYSTEM function provided by the Intel Fortran-90 compiler, which is an extension to ANSI standard Fortran-90.

Argument list:

(COMMAND, ERRORMSG)

Result:

(UTL_SYSTEM_FUNC)

Declarations for argument-list and result variables:

```
CHARACTER(LEN=*) , INTENT(IN)   :: COMMAND
CHARACTER(LEN=*) , INTENT(OUT)  :: ERRORMSG
LOGICAL                                     :: UTL_SYSTEM_FUNC
```

Explanation of arguments:

COMMAND is the operating-system command to be invoked.

ERRORMSG is the error message associated with the result of the SYSTEM command when the operating-system command is not executed successfully. If the operating-system command executes successfully, ERRORMSG is empty.

UTL_SYSTEM_FUNC is returned as .TRUE. if the operating-system command executes successfully and .FALSE. otherwise.

Parallel Processing (PLL) Module

Function PLL_CLOSE_AND_DELETE (added at JUPITER version 1.7.2)

Description: This integer function attempts to delete a file and verify that the file has been deleted.

Argument list:

(NUNIT, FNAME, MAXTRIES, WAITSECS)

Result:

(PLL_CLOSE_AND_DELETE)

Declarations for argument-list and result variables:

```
INTEGER,                                INTENT(IN) :: NUNIT
CHARACTER(LEN=*) ,                      INTENT(IN) :: FNAME
INTEGER, OPTIONAL,                      INTENT(IN) :: MAXTRIES
DOUBLE PRECISION, OPTIONAL, INTENT(IN) :: WAITSECS
```

Explanation of arguments:

NUNIT is a file unit number.

FNAME is a file name.

MAXTRIES is the maximum number of attempts that will be made to close the file. If

MAXTRIES is absent, a maximum of ten attempts will be made.

WAITSECS is a time interval, in seconds, between attempts to close the file. If WAITSECS is absent, the time between attempts is assigned as the module variable WAIT.

Notes:

1. If unit NUNIT is open, the file connected to unit NUNIT is closed, regardless whether the connected file is FNAME. If unit NUNIT is not open, it is used to connect to file FNAME, and then NUNIT is closed and file FNAME is deleted.
2. PLL_CLOSE_AND_DELETE returns 0 if the file is verified as having been deleted; it returns 1 otherwise.
3. If the file is not verified as having been deleted and GLOBAL_DATA module variable IVERB is greater than zero, a warning is written to the screen.

Function PLL_EXE_FUNC (added at JUPITER version 1.5.0)

Description: This logical function is similar to subroutine PLL_EXE in that it starts a model run and is intended to be called from a runner program. It differs from PLL_EXE in that it invokes UTL_SYSTEM_FUNC instead of UTL_SYSTEM and returns a .TRUE. or .FALSE. value and, in the event that the model command fails, an error message.

Argument list:

(IRUNRUNNER, NRUNRUNNER, ERRORMSG)

Result:

(PLL_EXE_FUNC)

Declarations for argument-list and result variables:

```
INTEGER,          INTENT( IN )   :: IRUNRUNNER
INTEGER,          INTENT( IN )   :: NRUNRUNNER
CHARACTER( LEN=* ), INTENT( OUT ) :: ERRORMSG
LOGICAL,          :: PLL_EXE_FUNC
```

Explanation of arguments:

IRUNRUNNER is the run number associated with the model run.

NRUNRUNNER is the number of model runs being made in parallel.

ERRORMSG is the error message associated with the result of invoking the model command when the model command is not executed successfully. If the model command executes successfully, ERRORMSG is empty.

Note: IRUNRUNNER and NRUNRUNNER are provided to the runner program in signal file "jdispar.rdy".