User's guide

Jérémie Gaidamour Pascal Hénon Yousef Saad

1 Overview

HIPS is a parallel solver for sparse linear (symmetric, unsymmetric, real or complex) systems. HIPS can be used as a stand-alone program that reads a sparse linear system from a file ; it also provides an interface to be called from any C, C++ or Fortran code. HIPS is developped by **Jérémie Gaidamour** and **Pascal Hénon** in the INRIA team-project "Scalapplix" in collaboration with **Yousef Saad** from the University of Minnesota.

2 Installation

2.1 Pre-requirements :

- C and Fortran compiler,
- an MPI-1 (Message Passing Interface) library,
- a BLAS (Basic Linear Algebra Subprograms) library,
- the latest SCOTCH 5.xx **or** METIS 4.0 library :
  - SCOTCH : http://www.labri.fr/perso/pelegrin/scotch/
  - METIS : http://glaros.dtc.umn.edu/gkhome/views/metis/

2.2 Installation procedure :

1. Install SCOTCH 5.xx or METIS 4.0.

2. Decompress the HIPS archive, go in the HIPS directory and copy from `Makefile_In_Example/` a `makefile.inc` corresponding to our architecture on the root of the HIPS installation directory. For example; if you are installing HIPS on a Linux system :

```
tar -xzf hips-*
cd hips-*
cp Makefile_Inc_Example/makefile.inc.linux ./makefile.inc
```

3. Edit the file `makefile.inc` which contains the compilation parameters :

   - Comment out one of these lines to compile the complex or the real version of HIPS :

     ```
     #COEFTYPE = -DTYPE_REAL
     #COEFTYPE = -DTYPE_COMPLEX
     ```

     On some architecture MPI has no complex type. In this case, you can compile HIPS with `COEFTYPE = -DTYPE_COMPLEX -DWITH_MPI_DOUBLE`.

   - If you are using METIS leave unchanged the line :

     ```
     PARTITIONER =
     ```

     Otherwise, set `PARTITIONER = -DSCOTCH_PART` to use SCOTCH.

     You can give the installation path of the partitionner library using the variables `METIS\_DIR` or `SCOTCH\_DIR`. Alternativly, you can directly set the library path (`LMETIS` or `LSCOTCH`) and the include path (`IMETIS` or `ISCOTCH`).

   - You can optionnaly define the integer sizes used in HIPS using the variable `INTSIZE`. Leave this variable unreferenced to use default C type of your machine. It is useful if :
     - you want to use HIPS in a Fortran code,
     - have 32 bits integers by default and encounter overflows (very large problems),
     - if your own code or library define specifically integer sizes.

4. Set the compiler and linker options specific to your installation :

   - Configure compilers and their flags (use for example `COPTFLAGS` and `FOPTFLAGS` to set optimizations flags like `-O3`).
   - Configure `LBLAS` variable to link to your BLAS library. For example, use :

     ```
     LBLAS = -lblas
     ```

     With the Intel Math Kernel Library (MKL), you will need something like :

     ```
     -L/mkl/10.0.3.020/lib/em64t -lmkl_intel_lp64    \
      -lmkl_sequential -lmkl_core -liomp5 -lpthread
     ```

     If you are using a multithreaded BLAS library (ATLAS, MKL, GotoBLAS, ...), disable this feathure (see manual of your BLAS library).

   - If your library path (`LD_LIBRARY_PATH`) are not properly set, you can add directory where are searched the libraries (variables `LSCOTCH`, `LMETIS`, `LMPI` and `LBLAS`). For example :

```
LBLAS = -L/path/to/libblas/ -lblas
```

- The search paths for header files can be controlled throught variables `ISCOTCH`, `IMETIS`, `IMPI` and `IBLAS`. For example, the following line will add `/path/to/mpi/include/` to the include path (to find `mpi.h`) :

  ```
  IMPI = -I/path/to/mpi/include/
  ```

5. Run `make all` to compile the library and the test programs. HIPS makefile are designed to compile with the GNU make : on certain architectures you will have to use `gmake` instead of `make` (`MAKE = gmake`).

3 How to run the hips test program

In `TESTS/PARALLEL` you should find an executable `testHIPS.ex`. The path of the matrix you want to try as well as some parameters must be set in the file `Inputs` which is in the same directory. Once you have set the parameters you want to test in `Inputs`; you can run HIPS like in this example (mpich with 16 processors) : `mpirun -np 16 ./testHIPS.ex <domsize>` where `<domsize>` is an integer that indicates which size of interior domain you would like to use. The `<domsize>` parameter should be small enough to create at least one domain per processor, otherwise the program will end with an error message. This parameter is optional in the case you choose the "ITERATIVE" (full iterative) strategy in the Inputs parameter file (see next subsection for details).

3.1 Input parameters of the test program

The test program testHIPS.ex read the input parameters in the file "Inputs". Here is a description of the Inputs file :

Example of an "Inputs" file for the hybrid method :

```
../MATRICES/bcsstk16.rsa   #0#  Matrix name and driver
2                          #1#  0=unsymmetric pattern, 1=symmetric pattern, 2=symmetric matrix in RS
0                          #2#  RHS file name (0=make a rhs (sol = [1]))
HYBRID                     #3#  Method (HYBRID, ITERATIVE)
1e-7                       #4#  Relative residual norm
ALL                        #5#  Fill-in : Strictly=0, Locally=ALL
100                        #6#  GMRES maximum iteration
100                        #7#  GMRES restart
0.00                       #8#  Numerical threshold in ILUT for interior domain
0.001                      #9#  Numerical threshold in ILUT for the interface
0.001                      #10# Numerical threshold for coupling between the interior level and Schu
5                          #11# Verbose level [0-4] (the higher the more it prints informations)
```

Important note for "HYBRID" method :

- you must set the parameter #8# to 0 (it means that an exact factorization is used for the interior domain matrix);
- the domsize parameter of "testHIPS.ex" is a domain average size ; it should be sufficiently small to ensure that at least one per processor can be created (the program will stop if it is not the case).

Example of an "Inputs" file for the full ITERATIVE method :

```
../MATRICES/bcsstk16.rsa   #0#  Matrix name and driver
2                          #1#  0=unsymmetric pattern, 1=symmetric pattern, 2=symmetric matrix in RS
0                          #2#  RHS file name (0=make a rhs (sol = [1]))
ITERATIVE                  #3#  Method (HYBRID, ITERATIVE)
1e-7                       #4#  Relative residual norm
ALL                        #5#  Fill-in : Strictly=0, Locally=ALL
100                        #6#  GMRES maximum iteration
100                        #7#  GMRES restart
0.001                      #8#  Numerical threshold in ILUT for interior domain
0.001                      #9#  Numerical threshold in ILUT for the interface
0.001                      #10# Numerical threshold for coupling between the interior level and Schu
5                          #11# Verbose level [0-4] (the higher the more it prints informations)
```

Important note for "ITERATIVE" method :

- it is better to set the parameter #8# a threshold value greater than 0 ;
- in the case of the full "ITERATIVE" strategy when no argument is given to testHIPS.ex, the parallelization scheme use one domain per processor.
  `mpirun -np 16 ./testHIPS.ex`

Example to read an unsymmetric matrix in Harwell boeing format (RUA) : If you are sure that the matrix non zero pattern is symmetric you can set the parameter #1# to 1. If it is not the case (or you are not sure) set this parameter to 0 : it will symmetrize the matrix non-zero pattern.

```
../MATRICES/orsirr_1.rua   #0#  Matrix name and driver
0                          #1#  0=unsymmetric pattern, 1=symmetric pattern, 2=symmetric matrix in RS
```

Example to read a symmetric complex matrix in matrix market format (require to compile the complex version of HIPS) :

```
3../MATRICES/young4c_.mtx   #0#  Matrix name and driver
2                          #1#  0=unsymmetric pattern, 1=symmetric pattern, 2=symmetric matrix in RS
```

Example to use a right hand side member : you must write in a file the values of the right hand side in **natural order** that is to say the original numbering of the matrix in inputs ; the i^{th} line of the file contains only one entry that is the value of the i^{th} component of the rhs.

```
./rhs                      #2#  RHS file name (0=make a rhs (sol = [1]))
```

3.2 Contents of HIPS directory

```
%  DOC/                    - documentation
  hips_user.pdf          - this file
  CeCILL-C_V1*.txt       - licensing agreement (english / french)

  SRC/                    - the library source files

  TESTS/                  - driver programs, input files
    MATRICES/             - sample problems in the Harwell-Boeing
                            format or Matrix Market
    PARALLEL/             - standalone program to solve a linear system
                            in parallel

  LIB/                    - directory in which the HIPS library and
                            HIPS headers are stored

  makefile.in             - compilers and compilation options are set
                            up in this file
  Makefile_In_Example/    - preconfigured makefile.in for some architecture.
  makefile                - complete makefile (does not need modification)
```

4 HIPS's Library Interface

4.1 Solver setup functions

Functions

- INTS HIPS_Initialize (INTS idnbr)
- INTS HIPS_SetDefaultOptions (INTS id, INTS stratnum)
- INTS HIPS_SetOptionINT (INTS id, INTS number, INTS value)
- INTS HIPS_SetOptionREAL (INTS id, INTS number, REAL value)

4.1.1 Function Documentation

4.1.1.1 INTS HIPS_Initialize (INTS *idnbr*)

In HIPS, each different linear problem is identified by an id. This id corresponds to a internal structure that contains intern data such as a matrix, options for a given problem. Unless you need to solve concurently several linear systems you do not need to set idnbr 1.

Parameters:

idnbr - Maximum number of different problems that will be created.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_INITIALIZE(IDNBR, IERROR)
  INTS, INTENT(IN)  :: IDNBR
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_INITIALIZE
```

4.1.1.2 INTS HIPS_SetDefaultOptions (INTS *id*, INTS *stratnum*)

Set default options corresponding to a specific solver strategy for the problem number ∗id∗. A strategy is identified by a ∗stratnum∗ ID. In HIPS, you can choose between a full iterative and a hybrid direct/iterative strategy.

Parameters:

id - Problem identification number.

stratnum - Solver strategy id. Can be set to HIPS_ITERATIVE or HIPS_-HYBRID.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error. This number can be used in HIPS_PrintError or HIPS_ExitOnError. Fortran interface:

```
SUBROUTINE HIPS_SETDEFAULTOPTIONS(ID, STRATNUM, IERROR)
  INTS, INTENT(IN)  :: ID, STRATNUM
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETDEFAULTOPTIONS
```

4.1.1.3 INTS HIPS_SetOptionINT (INTS *id*, INTS *number*, INTS *value*)

Set an option described by an integer number. An option is indentified by ∗number∗. ∗value∗ contain the value to assign to the option ∗number∗ for the problem ∗id∗.

Parameters:

id - Problem identification number.

number - Identification of the integer parameter.

value - Value to assign.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETOPTIONINT(ID, NUMBER, VALUE, IERROR)
  INTS, INTENT(IN)    :: ID, NUMBER, VALUE
  INTS, INTENT(OUT)   :: IERROR
END SUBROUTINE HIPS_SETOPTIONINT
```

4.1.1.4 INTS HIPS_SetOptionREAL (INTS *id*, INTS *number*, REAL *value*)

Set an option described by a real number. An option is indentified by ∗number∗. ∗value∗ contain the value to assign to the option ∗number∗ for the problem ∗id∗.

Parameters:

id - Problem identification number.

number - Identification of the integer parameter.

value - Value to set the parameter to.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError. Fortran interface:

```
SUBROUTINE HIPS_SETOPTIONREAL(ID, NUMBER, VALUE, IERROR)
  INTS, INTENT(IN)    :: ID, NUMBER
  REAL, INTENT(IN)    :: VALUE
  INTS, INTENT(OUT)   :: IERROR
END SUBROUTINE HIPS_SETOPTIONREAL
```

4.2 Graph setup function

Functions

- INTS HIPS_GraphBegin (INTS id, INTS n, INTL edgenbr)
- INTS HIPS_GraphEdge (INTS id, INTS col, INTS row)
- INTS HIPS_GraphEnd (INTS id)
- INTS HIPS_GraphDistrCSR (INTS id, INTS n, INTS ln, INTS ∗nodelist, INTL ∗lrowptr, INTS ∗cols)
- INTS HIPS_GraphGlobalCSR (INTS id, INTS n, INTL ∗rowptr, INTS ∗cols, INTS root)
- INTS HIPS_GraphGlobalCSC (INTS id, INTS n, INTL ∗colptr, INTS ∗rows, INTS root)
- INTS HIPS_GraphGlobalIJV (INTS id, INTS n, INTL nnz, INTS ∗row, INTS ∗col, INTS root)

4.2.1 Function Documentation

4.2.1.1 INTS HIPS_GraphBegin (INTS *id*, INTS *n*, INTL *edgenbr*)

Begin building the adjency graph for renumbering and all preprocessing.

Allocate temporary structures needed to build the graph.

Parameters:

id - Problem identification number.

n - Global number of nodes in the graph.

edgenbr - Number of edges which will be added in the graph by proc.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHBEGIN(ID, N, EDGENBR, IERROR)
  INTS,       INTENT(IN)  :: ID, N
  INTL,       INTENT(IN)  :: EDGENBR
  INTS,       INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHBEGIN
```

4.2.1.2 INTS HIPS_GraphDistrCSR (INTS *id*, INTS *n*, INTS *ln*, INTS ∗ *nodelist*, INTL ∗ *lrowptr*, INTS ∗ *cols*)

Enter the matrix adjacency graph using the distributed Compress Sparse Row format. Each processors has a set of row : the list of these rows (in global number) is in nodelist. If the set of rows

Parameters:

id - Problem identification number.

n - total number of vertice

ln - Number of local rows

nodelist - List of the local row in global numerbing : cols(lrowptr(i):lrowptr(i+1)) are the columns indices of edges in the row nodelist(i)

lrowptr - Index of the first element of each row in ∗LCOLS∗ and ∗VALUES∗ arrays.

lcols - Local column indice array.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHDISTRCSR(ID, N, LN, NODELIST, LROWPTR, COLS, IERROR)
  INTS,               INTENT(IN)  :: ID, LN, N
  INTS, DIMENSION(0), INTENT(IN)  :: NODELIST, COLS
  INTL, DIMENSION(0), INTENT(IN)  :: LROWPTR
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHDISTRCSR
```

4.2.1.3 INTS HIPS_GraphEdge (INTS *id*, INTS *col*, INTS *row*)

Adds an edge to the graph.

HIPS_GraphBegin must have been called before.

Parameters:

id - Problem identification number.

row - First vertex of the edge.

col - Second vertex of the edge.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHEDGE(ID, ROW, COL, IERROR)
  INTS,      INTENT(IN)  :: ID, ROW, COL
  INTS,      INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHEDGE
```

4.2.1.4 INTS HIPS_GraphEnd (INTS *id*)

End the graph building. HIPS_GraphBegin must have been called before.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHEND(ID, IERROR)
  INTS,      INTENT(IN)  :: ID
  INTS,      INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHEND
```

4.2.1.5 INTS HIPS_GraphGlobalCSC (INTS *id*, INTS *n*, INTL ∗ *colptr*, INTS ∗ *rows*, INTS *root*)

Build an adjency graph from a Compress Sparse Column matrix pattern.

Needs HIPS_SetDefaultOptions to be called before.

This function depends on integer parameter ∗HIPS_BASEVAL∗.

Parameters:

id - Problem identification number.

n - Global number of columns

colptr - Index of the first element of each column in ∗ROWS∗ array.

rows - Global row number array.

root - Root processor : this processor enter the global data.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHGLOBALCSC(ID, N, COLPTR, ROWS, ROOT, IERROR)
  INTS,               INTENT(IN)  :: ID, N, ROOT
  INTL, DIMENSION(0), INTENT(IN)  :: COLPTR
  INTL, DIMENSION(0), INTENT(IN)  :: ROWS
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHGLOBALCSC
```

4.2.1.6 INTS HIPS_GraphGlobalCSR (INTS *id*, INTS *n*, INTL ∗ *rowptr*, INTS ∗ *cols*, INTS *root*)

Build an adjency graph from a Compress Sparse Row matrix pattern.

Parameters:

id - Problem identification number.

n - Global number of columns

rowptr - Index of the first element of each row in ∗COLS∗ array.

cols - Global column numbers array.

root - Root processor : this processor enter the global data.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHGLOBALCSR(ID, N, ROWPTR, COLS, ROOT, IERROR)
  INTS,               INTENT(IN)  :: ID, N, ROOT
  INTL, DIMENSION(0), INTENT(IN)  :: ROWPTR
  INTS, DIMENSION(0), INTENT(IN)  :: COLS
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHGLOBALCSR
```

4.2.1.7 INTS HIPS_GraphGlobalIJV (INTS *id*, INTS *n*, INTL *nnz*, INTS * *row*, INTS * *col*, INTS *root*)

Build an adjency graph from a Compress Sparse Column matrix pattern.

Needs HIPS_SetDefaultOptions to be called before.

This function depends on integer parameter *HIPS_BASEVAL*.

Parameters:

id - Problem identification number.

n - Global number of unknowns.

nnz - Global number of non zeros.

row - Global column number array. edges.

col - Global row number array.

root - Root processor : this processor enter the global data.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GRAPHGLOBALIJV(ID, N, NNZ, ROW, COL, ROOT, IERROR)
  INTS,               INTENT(IN)  :: ID, N, ROOT
  INTL,               INTENT(IN)  :: NNZ
  INTS, DIMENSION(0), INTENT(IN)  :: ROW
  INTS, DIMENSION(0), INTENT(IN)  :: COL
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GRAPHGLOBALIJV
```

4.3 IO functions

Functions

- INTS HIPS_SetupSave (INTS id, char *directory)
- INTS HIPS_SetupLoad (INTS id, char *directory)
- INTS HIPS_LocalMatricesSave (INTS id, INTS nproc, INTS n, INTL *rowptr, INTS *col, COEF *values, char *directory)
- INTS HIPS_LocalMatriceLoad (INTS id, INTS *sym, INTS *n, INTL **rowptr, INTS **cols, COEF **values, char *sfile_path)

4.3.1 Detailed Description

Allows to save and load solver state after preprocessing.

4.3.2 Function Documentation

4.3.2.1 INTS HIPS_LocalMatriceLoad (INTS *id*, INTS * *sym*, INTS * *n*, INTL ** *rowptr*, INTS ** *cols*, COEF ** *values*, char * *sfile_path*)

This function loads the local matrix of a processor from disk. The local matrix must have been generated by HIPS_LocalMatricesSave.

Parameters:

id - Problem identification number.

sym - Return value : 0 unsymetric matrix, 1 symmetric matrix

n - Number of columns.

rowptr - Index of the first element of each rows in *COLS* and values* array.

cols - Row number array.

values - values array.

directory - Path to the directory where to save the local matrices.

In C the arrays rowptr and cols are allocated by the function. In Fortran, these array should have been allocated to a sufficient size. In Fortran, *STR_LEN* is the length of the string directory.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_LOCALMATRICELOAD(ID, SYM, N, ROWPTR, COLS, VALUES, DIRECTORY,
   STR_LEN, IERROR)
  INTS,              INTENT(IN)  :: ID, STR_LEN
  INTS,              INTENT(OUT)  :: SYM, N
  INTL, DIMENSION(0), INTENT(OUT)  :: ROWPTR
  INTS, DIMENSION(0), INTENT(OUT)  :: COLS
  COEF, DIMENSION(0), INTENT(OUT)  :: VALUES
  CHARACTER(len=*), INTENT(IN)  :: DIRECTORY
  INTS,              INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_LOCALMATRICELOAD
```

4.3.2.2 INTS HIPS_LocalMatricesSave (INTS *id*, INTS *nproc*, INTS *n*, INTL * *rowptr*, INTS * *col*, COEF * *values*, char * *directory*)

This function saves for each processor its local matrix. The local matrix is generated from the global one. Each matrix is save in a different file.

Parameters:

id - Solver instance identification number.

nproc - number of processors that will run the solver.

n - Number of columns.

rowptr - Index of the first element of each rows in ∗COLS∗ and values∗ array.

cols - Row number array.

values - values array.

directory - Path to the directory where to save the local matrices.

In Fortran, ∗STR_LEN∗ is the length of the string directory.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_LOCALMATRICESSAVE(ID, NPROC, N, ROWPTR, COLS, VALUES, DIRECTO
  RY, STR_LEN, IERROR)
  INTS,              INTENT(IN)  :: ID, NPROC, N, STR_LEN
  INTL, DIMENSION(0), INTENT(IN)  :: ROWPTR
  INTS, DIMENSION(0), INTENT(IN)  :: COLS
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  CHARACTER(len=*), INTENT(IN)  :: DIRECTORY
  INTS,              INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_LOCALMATRICESSAVE
```

4.3.2.3 INTS HIPS_SetupLoad (INTS *id*, char ∗ *directory*)

Loads preprocessing result from disk, into ∗directory∗, where it had been saved by HIPS_Save.

Parameters:

id - Problem identification number.

directory - Path to the directory where to load the solver preprocessing data.

In Fortran, ∗STR_LEN∗ is the length of the string directory.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETUPLOAD(ID, DIRECTORY, STR_LEN, IERROR)
  INTS,              INTENT(IN)  :: ID, STR_LEN
  CHARACTER(len=*), INTENT(IN)  :: DIRECTORY
  INTS,              INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETUPLOAD
```

4.3.2.4 INTS HIPS_SetupSave (INTS *id*, char * *directory*)

Save the result of the preprocessing steps (graph renumbering and partitioning) on disk in *directory*. Then computation can be resumed in parallel by using HIPS_Load.

To call this function, the graph of hte matrix must has been entered before by one of the HIPS graph function.

Parameters:

id - Solver instance identification number.

directory - Path to the directory where to save the preprocessing results.

In Fortran, *STR_LEN* is the length of the string directory.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETUPSAVE(ID, DIRECTORY, STR_LEN, IERROR)
  INTS,              INTENT(IN)  :: ID, STR_LEN
  CHARACTER(len=*), INTENT(IN)  :: DIRECTORY
  INTS,              INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETUPSAVE
```

4.4 Get the internal HIPS distribution

Functions

- INTS HIPS_GetLocalNodeNbr (INTS id, INTS *nodenbr)
- INTS HIPS_GetLocalNodeList (INTS id, INTS *nodelist)
- INTS HIPS_GetLocalUnknownNbr (INTS id, INTS *unkownnbr)
- INTS HIPS_GetLocalUnknownList (INTS id, INTS *unknownlist)
- INTS HIPS_SetPartition (INTS id, INTS ndom, INTS *mapptr, INTS *mapp)
- INTS HIPS_AssemblyBegin (INTS id, INTL nnz, INTS op, INTS op2, INTS mode, INTS symmetric)
- INTS HIPS_AssemblySetValue (INTS id, INTS row, INTS col, COEF value)
- INTS HIPS_AssemblySetNodeValues (INTS id, INTS row, INTS col, COEF *values)
- INTS HIPS_AssemblySetBlockValues (INTS id, INTS nrow, INTS *rowlist, INTS ncol, INTS *collist, COEF *values)
- INTS HIPS_AssemblyEnd (INTS id)
- INTS HIPS_MatrixReset (INTS id)
- INTS HIPS_FreePrecond (INTS id)

- INTS HIPS_MatrixLocalCSR (INTS id, INTS ln, INTS ∗unknownlist, INTL ∗lrowptr, INTS ∗lcols, COEF ∗values, INTS op, INTS op2, INTS sym)
- INTS HIPS_MatrixDistrCSR (INTS id, INTS ln, INTS ∗unknownlist, INTL ∗lrowptr, INTS ∗cols, COEF ∗values, INTS op, INTS op2, INTS mode, INTS sym)
- INTS HIPS_MatrixGlobalCSR (INTS id, INTS n, INTL ∗rowptr, INTS ∗cols, COEF ∗values, INTS root, INTS op, INTS sym)
- INTS HIPS_MatrixGlobalCSC (INTS id, INTS n, INTL ∗colptr, INTS ∗rows, COEF ∗values, INTS root, INTS op, INTS sym)
- INTS HIPS_MatrixGlobalIJV (INTS id, INTS n, INTL nnz, INTS ∗rows, INTS ∗cols, COEF ∗values, INTS root, INTS op, INTS sym)
- INTS HIPS_SetSubmatrixCoef (INTS id, INTS op, INTS op2, INTS n, INTL ∗ia, INTS ∗ja, COEF ∗a, INTS sym_matrix, INTS ln, INTS ∗nodelist)

4.4.1 Function Documentation

4.4.1.1 INTS HIPS_AssemblyBegin (INTS *id*, INTL *nnz*, INTS *op*, INTS *op2*, INTS *mode*, INTS *symmetric*)

Compute a partition of a graph with or without overlap between the partition. The overlap is computed using some heuristic that try to minimizes its size : it tries to compute an overlap of one node (or unknown). When no overlap is needed this function is merely a call to METIS or SCOTCH (graph partitioner).

Parameters:

ndom - Number of domains

overlap - 0 : no overlap in the partition , 1 : overlap as small as possible

numflag - 0 : numbering start from 0 (like in C), 1: numbering start from 1 (like in Fortran). This concern the inputs as well as the ouput of the function.

n - number of vertice in the graph.

rowptr - Index of the first element of each row in ∗COLS∗ array.

cols - Global column numbers array.

sym - 0 : the graph is not symmetric (it will be symmetrize in intern). 1 : the graph is symmetric (better is will be used as this inside the function).

mapptr - Array of indexes for domain in mapptr : mapp(mapptr(i):mapptr(i+1)-1) contains the nodes domain i

mapp - Array that contains the node lists for each domain.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
     Not provided.
*/
INTS HIPS_GraphPartition(INTS ndom, INTS overlap, INTS numflag, INTS n, INTL *row
      ptr, INTS *cols, INTS sym, INTS **mapptr, INTS **mapp);

INTS  HIPS_GetLocalDomainNbr(INTS id, INTS *domnbr, INTS *listsize);
INTS  HIPS_GetLocalDomainList(INTS id, INTS *mapptr, INTS *mapp);
```

4.4.1.2 INTS HIPS_AssemblyEnd (INTS *id*)

End an assembly loop.

HIPS_AssemblyBegin must have been called before.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_ASSEMBLYEND(ID, IERROR)
  INTS,      INTENT(IN)  :: ID
  INTS,      INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_ASSEMBLYEND
```

4.4.1.3 INTS HIPS_AssemblySetBlockValues (INTS *id*, INTS *nrow*, INTS * *rowlist*, INTS *ncol*, INTS * *collist*, COEF * *values*)

Set coefficients value for a dense submatrix (rowlist, colist).

Typically, this function is to be used to enter elementary matrix arising from PDE discretization.

HIPS_AssemblyBegin must have been called before.

Parameters:

id - Problem identification number.

nrow - Number of rows in the dense matrix.

rowlist - List of row indices (global ordering).

ncol - Number of columns in the dense matrix.

collist - List of column indices (global numbering).

values - Values array, stored by column (Fortran style)

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_ASSEMBLYSETBLOCKVALUES(ID, NROW, ROWLIST, &
                                & NCOL, COLLIST, VALUES, IERROR)
  INTS,               INTENT(IN)  :: ID, NROW, NCOL
  INTS, DIMENSION(0), INTENT(IN)  :: ROWLIST
  INTS, DIMENSION(0), INTENT(IN)  :: COLLIST
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_ASSEMBLYSETBLOCKVALUES
```

4.4.1.4 INTS HIPS_AssemblySetNodeValues (INTS *id*, INTS *row*, INTS *col*, COEF * *values*)

Set coefficients of a dense block corresponding to the edge (i, j) in the node graph. This function is useful only when the dof 1 (see the *HIPS_DOF* integer parameter) otherwise it is equivalent to HIPS_AssemblySetValue. Indeed each (i, j) entry of the node graph correspond to a (dof, dof) dense block in the linear system. The storage of the block is made by columns by columns in the vector values*.

HIPS_AssemblyBegin must have been called before.

Parameters:

id - Problem identification number.

row - Row index (global numbering) of the node.

col - Column index (global numbering) of the node.

values - Values of the dense block coefficient stored by columns.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_ASSEMBLYSETNODEVALUES(ID, ROW, COL, VALUES, IERROR)
  INTS,               INTENT(IN)  :: ID, ROW, COL
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_ASSEMBLYSETNODEVALUES
```

4.4.1.5 INTS HIPS_AssemblySetValue (INTS *id*, INTS *row*, INTS *col*, COEF *value*)

Set a coefficient value in the matrix.

HIPS_AssemblyBegin must have been called before.

Parameters:

id - Problem identification number.

row - Row index (global numbering) of the coefficient.

col - Column index (global numbering) of the coefficient.

value - Value of the coefficient.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_ASSEMBLYSETVALUE(ID, ROW, COL, VALUE, IERROR)
  INTS,       INTENT(IN)  :: ID, ROW, COL
  COEF,       INTENT(IN)  :: VALUE
  INTS,       INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_ASSEMBLYSETVALUE
```

4.4.1.6 INTS HIPS_FreePrecond (INTS *id*)

Free the preconditioner matrices.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_FREEPRECOND(ID, IERROR)
  INTS,       INTENT(IN)  :: ID
  INTS,       INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_FREEPRECOND
```

4.4.1.7 INTS HIPS_GetLocalNodeList (INTS *id*, INTS * *nodelist*)

This function gives the local node list corresponding to the HIPS domain partition.

The array *nodelist* must have been allocated with a size of at least *nodenbr* (obtained by the function HIPS_GetLocalNodeNbr).

Parameters:

id - Problem identification number.

nodelist - Array where to store the list of local nodes.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETLOCALNODELIST(ID, NODELIST, IERROR)
  INTS,              INTENT(IN)  :: ID
  ! Warning : 0 is not the size of the array.
  ! Writing DIMENSION(:) does not work with
  ! the C function call (fortran send the array size?)
  INTS, DIMENSION(0), INTENT(OUT) :: NODELIST
  INTS,              INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETLOCALNODELIST
```

4.4.1.8 INTS HIPS_GetLocalNodeNbr (INTS *id*, INTS * *nodenbr*)

This function gives the number of local nodes in the HIPS distribution. Be aware that the HIPS node partition uses an overlap (one node wide) between each domain.

The graph is required to get the distribution. It must has been entered by one of the HIPS graph input function.

Parameters:

id - Problem identification number.

nodenbr - Number of local nodes.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETLOCALNODENBR(ID, NODENBR, IERROR)
  INTS, INTENT(IN)  :: ID
  INTS, INTENT(OUT) :: NODENBR
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETLOCALNODENBR
```

4.4.1.9 INTS HIPS_GetLocalUnknownList (INTS *id*, INTS * *unknownlist*)

This function gives the local unknown list corresponding to the HIPS domain partition.

The array *unknownlist* must have been allocated with a size of at least *unknownnbr* (obtained by the function HIPS_GetLocalUnknownNbr). If the *HIPS_DOF* integer

option has been set to 1 (default value) then HIPS_GetLocalUnknownList is equivalent to HIPS_GetLocalNodeList.

Parameters:

id - Problem identification number.

unkownlist - Array where to store the list of local unknowns.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETLOCALUNKOWNLIST(ID, UNKOWNLIST, IERROR)
  INTS,               INTENT(IN)  :: ID
  INTS, DIMENSION(0), INTENT(OUT) :: UNKOWNLIST
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETLOCALUNKOWNLIST
```

4.4.1.10 INTS HIPS_GetLocalUnknownNbr (INTS *id*, INTS * *unkownnbr*)

This function gives the number of local unknowns in the HIPS distribution. Be aware that the HIPS unknowns partition uses an overlap between each domain. If the *HIPS_DOF* integer option has been set to 1 (default value) then HIPS_GetLocalUnknownNbr is equivalent to HIPS_GetLocalNodeNbr.

The graph is required to get the distribution. It must has been entered by one of the HIPS graph input function.

Parameters:

id - Problem identification number.

unkownnbr - Number of local unknowns.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETLOCALUNKOWNNBR(ID, UNKOWNNBR, IERROR)
  INTS, INTENT(IN)  :: ID
  INTS, INTENT(OUT) :: UNKOWNNBR
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETLOCALUNKOWNNBR
```

4.4.1.11 INTS HIPS_MatrixDistrCSR (INTS *id*, INTS *ln*, INTS * *unknownlist*, INTL * *lrowptr*, INTS * *cols*, COEF * *values*, INTS *op*, INTS *op2*, INTS *mode*, INTS *sym*)

Add a distrbuted matrix in Compress Sparse Row matrix to the matrix. Each processors has a set of row : the list of these rows (in global number) is in unknownlist. If the set of rows

IMPORTANT : the indices corresponds to the unknwon numbering, i.e. if you use HIPS_DOF 1, you have to expand the matrix in unknwon indices.

Parameters:

id - Problem identification number.

ln - Dimension of the local matrix

unknownlist - List of the local row in global numerbing : cols(lrowptr(i):lrowptr(i+1) are the columns indices of coefficients in row unknownlist(i))

lrowptr - Index of the first element of each row in *LCOLS* and *VALUES* arrays.

lcols - Local column indice array.

values - values array.

op - Operation to perform on the matrix problem : overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP). op2 - Operation to perform if several entries are entered for a same (i,j) location on different processor (see HIPS_ASSEMBLY_OP). This allows to control the operation to do for overlapped part of the matrix.

mode - Indicates if the user ensures he will respect the HIPS internal unknowns distribution (see HIPS_ASSEMBLY_MODE). Be careful : if you choose HIPS_ASSEMBLY_RESPECT, then any coefficient that is not in M(unknownlist1, unknownlist1) --where unknownlist1 is the HIPS internal distribution of the unknowns -- will be ignored. You can get the hips internal data distribution by using *HIPS_GetUnknownnbr* and *HIPS_GetUnknownlist*.

sym - Indicates if the user will give coefficients only in the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part. For a symmetric problem (indicated by the *HIPS_SYMMETRIC* integer option) any coefficient Aij = Aji so you should enter only the lower of upper triangular part of the matrix and set *sym* to 1.f

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXDISTRCSR(ID, LN, UNKNOWNLIST, LROWPTR, COLS, VALUES, &
                               & OP, OP2, MODE, SYM, IERROR)
  INTS,               INTENT(IN)  :: ID, LN, OP, OP2, SYM, MODE
  INTS, DIMENSION(0), INTENT(IN)  :: UNKNOWNLIST, COLS
  INTL, DIMENSION(0), INTENT(IN)  :: LROWPTR
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXDISTRCSR
```

4.4.1.12 INTS HIPS_MatrixGlobalCSC (INTS *id*, INTS *n*, INTL ∗ *colptr*, INTS ∗ *rows*, COEF ∗ *values*, INTS *root*, INTS *op*, INTS *sym*)

Add the given global Compress Sparse Column matrix to the matrix.

Parameters:

id - Problem identification number.

n - Number of columns.

colptr - Index of the first element of each column in ∗ROWS∗ and values∗ array.

rows - Row number array.

values - values array.

root - Root processor for MPI communications.

op - Operation to perform on the matrix problem : overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP).

sym - Indicates if the user will give coefficients only in the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part. For a symmetric problem (indicated by the ∗HIPS_SYMMETRIC∗ integer option) any coefficient Aij = Aji so you should enter only the lower of upper triangular part of the matrix and set ∗sym∗ to 1.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXGLOBALCSC(ID, N, COLPTR, ROWS, &
                                & VALUES, ROOT, OP, SYM, IERROR)
  INTS,               INTENT(IN)  :: ID, N, ROOT, OP, SYM
  INTL, DIMENSION(0), INTENT(IN)  :: COLPTR
  INTS, DIMENSION(0), INTENT(IN)  :: ROWS
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXGLOBALCSC
```

4.4.1.13 INTS HIPS_MatrixGlobalCSR (INTS *id*, INTS *n*, INTL ∗ *rowptr*, INTS ∗ *cols*, COEF ∗ *values*, INTS *root*, INTS *op*, INTS *sym*)

Add the given global Compress Sparse Row matrix to the matrix.

Parameters:

id - Problem identification number.

n - Number of columns.

rowptr - Index of the first element of each row in ∗COLS∗ and values∗ array.

cols - Column number array.

values - values array.

root - Root processor for MPI communications.

op - Operation to perform on the matrix problem : overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP).

sym - Indicates if the user will give coefficients only in the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part. For a symmetric problem (indicated by the ∗HIPS_SYMMETRIC∗ integer option) any coefficient Aij = Aji so you should enter only the lower of upper triangular part of the matrix and set ∗sym∗ to 1.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXGLOBALCSR(ID, N, ROWPTR, COLS, VALUES, &
                                & ROOT, OP, SYM,  IERROR)
  INTS,               INTENT(IN)  :: ID, N, ROOT, OP, SYM
  INTL, DIMENSION(0), INTENT(IN)  :: ROWPTR
  INTS, DIMENSION(0), INTENT(IN)  :: COLS
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXGLOBALCSR
```

4.4.1.14 INTS HIPS_MatrixGlobalIJV (INTS *id*, INTS *n*, INTL *nnz*, INTS ∗ *rows*, INTS ∗ *cols*, COEF ∗ *values*, INTS *root*, INTS *op*, INTS *sym*)

Add the given global Compress Sparse Column matrix to the matrix.

Parameters:

id - Problem identification number.

n - Number of edges.

nnz - Number of non zeros.

rows - Global row number array.

cols - Global column number array.

values - values array.

root - Root processor for MPI communications.

op - Operation to perform on the matrix problem : overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP).

sym - Indicates if the user will give coefficients only in the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part. For a symmetric problem (indicated by the ∗HIPS_SYMMETRIC∗ integer option) any coefficient Aij = Aji so you should enter only the lower of upper triangular part of the matrix and set ∗sym∗ to 1.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXGLOBALIJV(ID, N, NNZ, ROWS, COLS, VALUES, &
                                & ROOT, OP, SYM, IERROR)
  INTS,               INTENT(IN)  :: ID, ROOT, OP, SYM, N
  INTL,               INTENT(IN)  :: NNZ
  INTS, DIMENSION(0), INTENT(IN)  :: ROWS, COLS
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXGLOBALIJV
```

4.4.1.15 INTS HIPS_MatrixLocalCSR (INTS *id*, INTS *ln*, INTS ∗ *unknownlist*, INTL ∗ *lrowptr*, INTS ∗ *lcols*, COEF ∗ *values*, INTS *op*, INTS *op2*, INTS *sym*)

Add a submatrix in Compress Sparse Row matrix to the matrix. The submatrix M corresponds to a matrix (unknownlist, unknownlist) in CSR format where unknownlist is a subset of the list of unknowns given by the function ∗HIPS_GetLocalUnknownList∗. The submatrix is locally numbered : i.e. M(i, i) is to be added in A(nodelist, nodelist).

IMPORTANT : the indices corresponds to the unknwon numbering, i.e. if you use HIPS_DOF 1, you have to expand the matrix in unknwon indices.

Parameters:

id - Problem identification number.

ln - Dimension of the local matrix

unknownlist - List of the local unknown in global numbering

lrowptr - Index of the first element of each row in ∗LCOLS∗ and ∗VALUES∗ arrays.

lcols - Local column indice array.

values - values array.

op - Operation to perform on the matrix problem : overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP).

op2 - Operation to perform if several entries are entered for a same (i,j) location on different processor (see HIPS_ASSEMBLY_OP). This allows to control the operation to do for overlapped part of the matrix.

sym - Indicates if the user will give coefficients only in the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part. For a symmetric problem (indicated by the ∗HIPS_SYMMETRIC∗ integer option) any coefficient Aij = Aji so you should enter only the lower of upper triangular part of the matrix and set ∗sym∗ to 1.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXLOCALCSR(ID, LN, UNKNOWNLIST, LROWPTR, LCOLS, VALUES, &

                               & OP, OP2, SYM, IERROR)
  INTS,               INTENT(IN)  :: ID, LN, OP, OP2, SYM
  INTS, DIMENSION(0), INTENT(IN)  :: UNKNOWNLIST, LCOLS
  INTL, DIMENSION(0), INTENT(IN)  :: LROWPTR
  COEF, DIMENSION(0), INTENT(IN)  :: VALUES
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXLOCALCSR
```

4.4.1.16 INTS HIPS_MatrixReset (INTS *id*)

Reset the matrix structure.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXRESET(ID, IERROR)
  INTS,       INTENT(IN)  :: ID
  INTS,       INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXRESET
```

4.4.1.17 INTS HIPS_SetPartition (INTS *id*, INTS *ndom*, INTS ∗ *mapptr*, INTS ∗ *mapp*)

Sets a partition defined by the user. It can only be used for the recursive ITERATIVE strategy.

Parameters:

id - Problem identification number.

ndom - Number of domains

mapptr - Array of indexes for domain in mapptr : mapp(mapptr(i):mapptr(i+1)-1) contains the nodes domain i

mapp - Array that contains the node lists for each domain.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETPARTITION(ID, NDOM, MAPPTR, MAPP, IERROR)
  INTS, INTENT(IN)    :: ID, NDOM
  INTS, DIMENSION(0), INTENT(IN)  ::  MAPPTR
  INTS, DIMENSION(0), INTENT(IN)  ::  MAPP
  INTS, INTENT(OUT)   :: IERROR
END SUBROUTINE HIPS_SETPARTITION
```

4.4.1.18 INTS HIPS_SetSubmatrixCoef (INTS *id*, INTS *op*, INTS *op2*, INTS *n*, INTL * *ia*, INTS * *ja*, COEF * *a*, INTS *sym_matrix*, INTS *ln*, INTS * *nodelist*)

4.5 Fill the right-hand-side member

Functions

- INTS HIPS_SetGlobalRHS (INTS id, COEF *b, INTS proc_root, INTS op)
- INTS HIPS_SetLocalRHS (INTS id, COEF *b, INTS op, INTS op2)
- INTS HIPS_SetRHS (INTS id, INTS unknownnbr, INTS *unknownlist, COEF *b, INTS op, INTS op2, INTS mode)
- INTS HIPS_RHSReset (INTS id)

4.5.1 Function Documentation

4.5.1.1 INTS HIPS_RHSReset (INTS *id*)

Reset the right-hand-side.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_RHSRESET(ID, IERROR)
  INTS,      INTENT(IN)  :: ID
  INTS,      INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_RHSRESET
```

4.5.1.2 INTS HIPS_SetGlobalRHS (INTS *id*, COEF * *b*, INTS *proc_root*, INTS *op*)

Set the right-hand-side member in global mode.

Parameters:

id - Problem identification number.

b - Array of size global column number which correspond to the right-hand-side member.

op - Operation on the right hand side : overwrite values of the existing rhs or add new entries to the former ones (see HIPS_ASSEMBLY_OP).

root - Indicates which processor enter the global right-hand-side member,

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETGLOBALRHS(ID, B, ROOT, OP, IERROR)
  INTS,               INTENT(IN)  :: ID, ROOT, OP
  COEF, DIMENSION(0), INTENT(IN)  :: B
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETGLOBALRHS
```

4.5.1.3 INTS HIPS_SetLocalRHS (INTS *id*, COEF * *b*, INTS *op*, INTS *op2*)

Set the right-hand-side member in local mode.

Parameters:

id - Problem identification number.

b - Array of size local column number which correspond to the right-hand-side member.

op - overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP). op2 - Operation to perform if several entries

are entered for a same vector component on different processor (see HIPS_-ASSEMBLY_OP). This allows to control the operation to do for overlapped part of the rhs.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETLOCALRHS(ID, B, OP, OP2,  IERROR)
  INTS,               INTENT(IN)  :: ID, OP, OP2
  COEF, DIMENSION(0), INTENT(IN)  :: B
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETLOCALRHS
```

4.5.1.4 INTS HIPS_SetRHS (INTS *id*, INTS *unknownnbr*, INTS * *unknownlist*, COEF * *b*, INTS *op*, INTS *op2*, INTS *mode*)

Set the right-hand-side member, giving the list of coefficient that we set.

mode* shouldn't be *HIPS_ASSEMBLY_RESPECT* if neither HIPS_-GetLocalNodeList nor HIPS_GetLocalUnknownList has been called.

Parameters:

id - Problem identification number.

n - Number of coefficients to set.

coefsidx - List of global index of the coefficients to set.

B - Array of coefficients values.

op - overwrite values of the existing matrix or add new entries to the former ones (see HIPS_ASSEMBLY_OP). op2 - Operation to perform if several entries are entered for a same vector component on different processor (see HIPS_-ASSEMBLY_OP). This allows to control the operation to do for overlapped part of the rhs.

mode - Indicates if user ensure he will respect solvers distribution (see HIPS_-ASSEMBLY_MODE).

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETRHS(ID, N, COEFSIDX, B, OP, OP2, MODE, IERROR)
  INTS,               INTENT(IN)  :: ID, N, OP, OP2, MODE
  INTS, DIMENSION(0), INTENT(IN)  :: COEFSIDX
  COEF, DIMENSION(0), INTENT(IN)  :: B
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETRHS
```

4.6 Get the solution

Functions

- INTS HIPS_GetGlobalSolution (INTS id, COEF ∗x, INTS root)
- INTS HIPS_GetLocalSolution (INTS id, COEF ∗x)
- INTS HIPS_GetSolution (INTS id, INTS n, INTS ∗nlist, COEF ∗x, INTS mode)

4.6.1 Function Documentation

4.6.1.1 INTS HIPS_GetGlobalSolution (INTS *id*, COEF ∗ *x*, INTS *root*)

Perform Factorization and Solve, if needed, and then fill the global solution in ∗x∗.

Parameters:

id - Problem identification number.

x - Array of size global column number which will contain the solution

root - Indicates which processor will have the solution at the end of the call, -1 for all.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETGLOBALSOLUTION(ID, X, ROOT, IERROR)
  INTS,               INTENT(IN)  :: ID, ROOT
  COEF, DIMENSION(0), INTENT(OUT) :: X
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETGLOBALSOLUTION
```

4.6.1.2 INTS HIPS_GetLocalSolution (INTS *id*, COEF ∗ *x*)

Perform Factorization and Solve, if needed, and then fill the local solution in ∗x∗.

Parameters:

id - Problem identification number.

x - Array that will contain the local solution corresponding to the HIPS unknown distribution (that can be obtained by HIPS_GetUnknownList).

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETLOCALSOLUTION(ID, X, IERROR)
  INTS,               INTENT(IN)  :: ID
  COEF, DIMENSION(0), INTENT(OUT) :: X
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETLOCALSOLUTION
```

4.6.1.3 INTS HIPS_GetSolution (INTS *id*, INTS *n*, INTS * *nlist*, COEF * *x*, INTS *mode*)

Perform Factorization and Solve, if needed, and then fill the solution in *x* followin the given index list.

Parameters:

id - Problem identification number.

n - Number of coefficients user wants to get.

coefsidx - List of the coefficients user wants to get.

x - Array that contain the local part of solution in return.

mode - Indicates if the user is sure to respect the distribution.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETSOLUTION(ID, N, COEFSIDX, X, MODE, IERROR)
  INTS,               INTENT(IN)  :: ID, MODE, N
  INTS, DIMENSION(0), INTENT(IN)  :: COEFSIDX
  COEF, DIMENSION(0), INTENT(OUT) :: X
  INTS,               INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETSOLUTION
```

4.7 Clean up

Functions

- INTS HIPS_Clean (INTS id)
- INTS HIPS_Finalize ()

4.7.1 Function Documentation

4.7.1.1 INTS HIPS_Clean (INTS *id*)

Clean the given instance of the solver structure's.

Parameters:

id - Problem identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_CLEAN(ID, IERROR)
  INTS, INTENT(IN)  :: ID
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_CLEAN
```

4.7.1.2 INTS HIPS_Finalize ()

Clean all not cleaned instances and instances ID array.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_FINALIZE(IERROR)
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_FINALIZE
```

4.8 Get HIPS's Infos

Functions

- INTS HIPS_GetInfoINT (INTS id, INTS infonum, INTL ∗value)
- INTS HIPS_GetInfoREAL (INTS id, INTS infonum, REAL ∗value)
- void HIPS_PrintError (INTS ierror)
- void HIPS_ExitOnError (INTS ierror)

4.8.1 Function Documentation

4.8.1.1 void HIPS_ExitOnError (INTS *ierror*)

Print the error message corresponding to ierror. If the ierr is not HIPS_SUCCESS then the program is stopped.

Parameters:

ierror - Error identification number.

void

Fortran interface:

```
SUBROUTINE HIPS_EXITONERROR(IERROR)
  INTS, INTENT(IN) :: IERROR
END SUBROUTINE HIPS_EXITONERROR
```

4.8.1.2 INTS HIPS_GetInfoINT (INTS *id*, INTS *infonum*, INTL * *value*)

Get an info (integer number) from HIPS.

See HIPS_INFO_INT and the solver documentation to get available info list.

Parameters:

id - Problem identification number.

infonum - Wanted information number.

value - Integer which will contain the value of the information.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETINFOINT(ID, INFONUM, VALUE, IERROR)
 INTS, INTENT(IN)  :: ID, INFONUM
 INTL, INTENT(OUT) :: VALUE
 INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETINFOINT
```

4.8.1.3 INTS HIPS_GetInfoREAL (INTS *id*, INTS *infonum*, REAL * *value*)

Get an info (real number) from HIPS.

See HIPS_INFO_REAL and the solver documentation to get available info list.

Parameters:

id - Problem identification number.

infonum - Wanted information number.

value - Integer which will contain the value of the information.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_GETINFOREAL(ID, INFONUM, VALUE, IERROR)
 INTS, INTENT(IN)  :: ID, INFONUM
 REAL, INTENT(OUT) :: VALUE
 INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_GETINFOREAL
```

4.8.1.4 void HIPS_PrintError (INTS *ierror*)

Print the error message corresponding to ierror

Parameters:

ierror - Error identification number.

void

Fortran interface:

```
SUBROUTINE HIPS_PRINTERROR(IERROR)
  INTS, INTENT(IN):: IERROR
END SUBROUTINE HIPS_PRINTERROR
```

4.9 HIPS advanced functions

Functions

- INTS HIPS_SetCommunicator (INTS id, MPI_Comm mpicom)
- INTS HIPS_MatrixVectorProduct (INTS id, COEF ∗x, COEF ∗y)
- INTS HIPS_TransposeMatrix (INTS id)
- INTS HIPS_ReadOptionsFromFile (INTS id, char ∗inputsname, INTS ∗sym_pattern, INTS ∗sym_matrix, char ∗matrixname, char ∗rhsname)
- INTS HIPS_CheckSolution (INTS id, INTS n, INTL ∗rowptr, INTS ∗cols, COEF ∗values, COEF ∗sol, COEF ∗rhs, INTS sym)
- INTS HIPS_GetSubmatrix (INTS ln, INTS numflag, INTS ∗nodelist, INTS n, INTL ∗ia, INTS ∗ja, COEF ∗a, INTL ∗∗lia, INTS ∗∗lja, COEF ∗∗la)
- void Matrix_Read (INTS job, INTS ∗t_n, INTL ∗t_nnz, INTL ∗ia, INTS ∗ja, COEF ∗a, INTS ∗sym_matrix, char ∗matrix)

4.9.1 Function Documentation

4.9.1.1 INTS HIPS_CheckSolution (INTS *id*, INTS *n*, INTL ∗ *rowptr*, INTS ∗ *cols*, COEF ∗ *values*, COEF ∗ *sol*, COEF ∗ *rhs*, INTS *sym*)

This function check if the global solution (in initial numbering) obtained by HIPS is exact (i.e. respect the relative residual norm error used to stop the

convergence in HIPS). It multiplies the solution by the initial matrix in CSR and check the error compared to the expected error

Parameters:

id - Problem identification number.

n - Number of rows.

rowptr - Index of the first element of each row in ∗COLS∗ and values∗ array.

cols - Column number array.

values - values array.

sym - Indicates if the crs represents only the lower triangular part and that HIPS must consider that these coefficients are also the same in the upper triangular part.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

SUBROUTINE HIPS_CHECKSOLUTION(ID, N, ROWPTR, COLS, VALUES, SOL, RHS, SYM, IERROR) INTS, INTENT(IN) :: ID, N, SYM INTL, DIMENSION(0), INTENT(IN) :: ROWPTR INTS, DIMENSION(0), INTENT(IN) :: COLS COEF, DIMENSION(0), INTENT(IN) :: VALUES COEF, DIMENSION(0), INTENT(IN) :: SOL COEF, DIMENSION(0), INTENT(IN) :: RHS INTS, INTENT(OUT) :: IERROR END SUBROUTINE HIPS_CHECKSOLUTION

4.9.1.2 INTS HIPS_GetSubmatrix (INTS *ln*, INTS *numflag*, INTS ∗ *nodelist*, INTS *n*, INTL ∗ *ia*, INTS ∗ *ja*, COEF ∗ *a*, INTL ∗∗ *lia*, INTS ∗∗ *lja*, COEF ∗∗ *la*)

4.9.1.3 INTS HIPS_MatrixVectorProduct (INTS *id*, COEF ∗ *x*, COEF ∗ *y*)

This function does a the product y = A.c where A is the matrix of the problem id.

Parameters:

id - Problem identification number.

x - Array that contain the local part of x corresponding to the HIPS unknown distribution (that can be obtained by HIPS_GetUnknownList).

y - Array that contain the local part of y corresponding to the HIPS unknown distribution (that can be obtained by HIPS_GetUnknownList).

ierror - Error identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_MATRIXVECTORPRODUCT(ID, X, Y, IERROR)
  INTS, INTENT(IN) :: ID
  COEF, DIMENSION(0), INTENT(IN) :: X
  COEF, DIMENSION(0), INTENT(OUT) :: Y
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_MATRIXVECTORPRODUCT
```

4.9.1.4 INTS HIPS_ReadOptionsFromFile (INTS *id*, char * *inputsname*, INTS * *sym_pattern*, INTS * *sym_matrix*, char * *matrixname*, char * *rhsname*)

This function reads the parameter in a "Inputs" file. This function can be used to write a code that load a matrix from disk and/or should read some HIPS basic parameters from a file instead of using HIPS_SetOptionINT or HIPS_OptionREAL in the code.

Parameters:

id - Problem identification number.

inputsname - name of the "inputs" file. sym_pattern - return wether the matrix has a symmetric non zero pattern (1) or not (0) sym_matrix - return wether the matrix is symmetric (only lower triangular is stored in CSC) or not

matrixname - return name of the matrix file.

rhsname - return name of the rhs file

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

SUBROUTINE HIPS_READOPTIONSFROMFILE(ID, SYM_PATTERN, SYM_-MATRIX, INPUTSNAME, MATRIXNAME, RHSNAME, IERROR) INTS, IN-TENT(IN) :: ID INTS, INTENT(OUT) :: SYM_PATTERN, SYM_MATRIX CHARACTER(LEN=*), INTENT(IN) :: INPUTSNAME CHARACTER(LEN=*), INTENT(OUT):: MATRIXNAME, RHSNAME INTS, INTENT(OUT):: IERROR END SUBROUTINE HIPS_READOPTIONSFROMFILE

4.9.1.5 INTS HIPS_SetCommunicator (INTS *id*, MPI_Comm *mpicom*)

Sets MPI communicator for the given solver instance.

Needs HIPS_SetDefaultOptions to be called before to initiate solver instance data.

Musn't be called before HIPS_SAVE, HIPS_LOAD, HIPS_GetLocalNodeNbr nor HIPS_GetLocalUnknownNbr because the solver as to be runned with the same MPI communicator all along.

If this function is not called, MPI communicator will be MPI_COMM_WORLD∗.

This function may not exist if the solver has been compiled without MPI.

Parameters:

id - Problem identification number.

mpicomm - MPI communicator to be used for solving this problem.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_SETCOMMUNICATOR(ID, MPICOMM, IERROR)
  INTS,      INTENT(IN)  :: ID
  INTEGER,  INTENT(IN)  :: MPICOMM
  INTS,      INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_SETCOMMUNICATOR
```

4.9.1.6 INTS HIPS_TransposeMatrix (INTS *id*)

This function transpose the matrix problem.

Parameters:

id - Problem identification number.

ierror - Error identification number.

HIPS_SUCCESS - Successful return. HIP_ERR_xx - A number that corresponds to a specific error that can be used with HIPS_PrintError or HIPS_ExitOnError.

Fortran interface:

```
SUBROUTINE HIPS_TRANSPOSEMATRIX(ID, IERROR)
  INTS, INTENT(IN) :: ID
  INTS, INTENT(OUT) :: IERROR
END SUBROUTINE HIPS_TRANSPOSEMATRIX
```

4.9.1.7 void Matrix_Read (INTS *job*, INTS ∗ *t_n*, INTL ∗ *t_nnz*, INTL ∗ *ia*, INTS ∗ *ja*, COEF ∗ *a*, INTS ∗ *sym_matrix*, char ∗ *matrix*)

Read a matrix from file. This function is needed tby the hips test example in TEST/PARALLEL/

SUBROUTINE MATRIX_READ(job, t_n, t_nnz, ia, ja, a, sym_matrix, matrix) INTS :: job INTS :: t_n INTL :: t_nnz INTL, DIMENSION(0) :: ia INTS, DIMENSION(0) :: ja COEF, DIMENSION(0) :: a INTS :: sym_matrix CHARACTER(LEN=200) :: matrix END SUBROUTINE MATRIX_READ

4.10 HIPS's constants

Enumerations

- enum HIPS_STRATNUM {

  HIPS_DIRECT = 0, HIPS_ILUT = 1, HIPS_ITERATIVE = 1, HIPS_HYBRID = 2,

  HIPS_BLOCK = 3 }
- enum HIPS_IPARAM {

  HIPS_SYMMETRIC = 0, HIPS_VERBOSE = 1, HIPS_SCALE = 2, HIPS_LOCALLY = 3,

  HIPS_KRYLOV_RESTART = 4, HIPS_ITMAX = 5, HIPS_FORWARD = 6, HIPS_SCHUR_METHOD = 7,

  HIPS_ITMAX_SCHUR = 8, HIPS_PARTITION_TYPE = 9, HIPS_KRYLOV_METHOD = 10, HIPS_DOMSIZE = 11,

  HIPS_SMOOTH_ITER_RATIO = 12, HIPS_DOMNBR = 13, HIPS_REORDER = 14, HIPS_SCALENBR = 15,

  HIPS_MASTER = 16, HIPS_COARSE_GRID = 17, HIPS_CHECK_GRAPH = 18, HIPS_CHECK_MATRIX = 19,

  HIPS_DUMP_CSR = 20, HIPS_IMPROVE_PARTITION = 21, HIPS_TAGNBR = 22, HIPS_SHIFT_DIAG = 23,

  HIPS_GRAPH_SYM = 24, HIPS_GRID_DIM = 25, HIPS_GRID_3D = 26, HIPS_DISABLE_PRECOND = 27,

  HIPS_FORTRAN_NUMBERING = 28, HIPS_DOF = 29, HIPS_PIVOTING = 30 }
- enum HIPS_RPARAM {

  HIPS_PREC = 0, HIPS_DROPTOL0 = 1, HIPS_DROPTOL1 = 2, HIPS_DROPSCHUR = 3,

  HIPS_DROPTOLE = 4, HIPS_AMALG = 5 }
- enum HIPS_RETURNS {

  HIPS_SUCCESS = 1, HIPS_ERR_ALLOCATE, HIPS_ERR_IO, HIPS_ERR_PARAMETER,

  HIPS_ERR_MATASSEMB, HIPS_ERR_RHSASSEMB, HIPS_ERR_PARASETUP, HIPS_ERR_CALL,

  HIPS_ERR_PRECOND, HIPS_ERR_SOLVE, HIPS_ERR_KRYLOV, HIPS_ERR_CHECK }

- enum HIPS_INFO_INT {

  HIPS_INFO_NNZ = 0, HIPS_INFO_NNZ_PEAK, HIPS_INFO_DIM, HIPS_-INFO_OUTER_ITER,

  HIPS_INFO_INNER_ITER, HIPS_INFO_ITER }
- enum HIPS_INFO_REAL {

  HIPS_INFO_PRECOND_TIME = 0, HIPS_INFO_SOLVE_TIME, HIPS_-INFO_FILL, HIPS_INFO_FILL_PEAK,

  HIPS_INFO_RES_NORM }
- enum HIPS_ASSEMBLY_MODE { HIPS_ASSEMBLY_RESPECT, HIPS_-ASSEMBLY_FOOL }
- enum HIPS_ASSEMBLY_OP { HIPS_ASSEMBLY_OVW = 0, HIPS_-ASSEMBLY_ADD }

4.10.1 Enumeration Type Documentation

4.10.1.1 enum HIPS_ASSEMBLY_MODE

Enum: HIPS_ASSEMBLY_MODE

Indicates if user can ensure that the information he is giving respects the solver distribution.

HIPS_ASSEMBLY_RESPECT - User ensure he respects distribution during assembly. HIPS_ASSEMBLY_FOOL - User is not sure he will respect ditribution during assembly

Enumerator:

HIPS_ASSEMBLY_RESPECT

HIPS_ASSEMBLY_FOOL

4.10.1.2 enum HIPS_ASSEMBLY_OP

Enum: HIPS_ASSEMBLY_OP

Operations possible when a coefficient appear twice.

HIPS_ASSEMBLY_OVW - Coefficients will be overwriten during assembly. HIPS_-ASSEMBLY_ADD - Coefficients will be added to the matrix. Do not sum overlaped values between processors.

Enumerator:

HIPS_ASSEMBLY_OVW

HIPS_ASSEMBLY_ADD

4.10.1.3 enum HIPS_INFO_INT

Enum: HIPS_INFO_INT

HIPS integer information identifiers.

Contains: HIPS_INFO_NNZ - Number of non zeros stored in the preconditioner (end of the preconditioning step). HIPS_INFO_NNZ_PEAK - Maximum number (peak) of non-zeros stored during preconditioning step. HIPS_INFO_DIM - Dimension of the global matrix HIPS_INFO_OUTER_ITER, - Number of outer iterations HIPS_INFO_-INNER_ITER, - Number of inner iterations (Schur complement) HIPS_INFO_ITER - Number of iterations (inner for hybrid, outer for iterative)

Enumerator:

HIPS_INFO_NNZ

HIPS_INFO_NNZ_PEAK

HIPS_INFO_DIM

HIPS_INFO_OUTER_ITER

HIPS_INFO_INNER_ITER

HIPS_INFO_ITER

4.10.1.4 enum HIPS_INFO_REAL

Enum: HIPS_INFO_REAL

HIPS integer information identifiers.

Contains: HIPS_INFO_PRECOND_TIME - Preconditioning time. HIPS_INFO_-SOLVE_TIME - Solving time (total time in the preconditioned Krylov method). HIPS_INFO_FILL - Fill-in (ratio with the number of non-zeros in the initial matrix) in the preconditioner. HIPS_INFO_FILL_PEAK - Maximum fill-in (ratio with the number of non-zeros in the initial matrix) during the preconditioning step. HIPS_-INFO_RES_NORM - The relative residual norm achieved in the last resolution

Enumerator:

HIPS_INFO_PRECOND_TIME

HIPS_INFO_SOLVE_TIME

HIPS_INFO_FILL

HIPS_INFO_FILL_PEAK

HIPS_INFO_RES_NORM

4.10.1.5 enum HIPS_IPARAM

Enum: HIPS_IPARAM

Hips integer parameters identifiers.

Solvers may implement is own list of parameters. Contains: HIPS_SYMMETRIC - [0,1] (default 0) this means that the matrix is symmetric and that you want to use the symmetric algorithms of HIPS (it saves half the computation and memory). { In this case, only the upper triangular part of the CSR matrix given as input to HIPS functions are considered}. HIPS_VERBOSE - [0-5] (default 2) level of informations printed in HIPS functions (0 the less information)

HIPS_SCALE - HIPS developers reserved. HIPS_LOCALLY - [0-] number of level that use the HIPS locally consistent fill-in pattern. This option defines the fill-in pattern (structurally and not numerically as in ILUT In general one will use 0 (no fill-in outside the original diagonal block pattern) or a high value (100 for example) to allow the fill-in anywhere in the Schur complement pattern. HIPS_KRYLOV_RESTART - restart parameter of GMRES. HIPS_ITMAX - maximum number of iteration allowed in the krylov method. In the full "ITERATIVE" mode (see ∗HIPS_STRATNUM∗), you can set "-1" to make a simple forward/backward substitution (for example if you want to use HIPS as a preconditioner in your method). HIPS_FORWARD - HIPS developers reserved. HIPS_SCHUR_METHOD - HIPS developers reserved. HIPS_ITMAX_SCHUR - HIPS_PARTITION_TYPE - HIPS developers reserved. HIPS_KRYLOV_METHOD - 0 Preconditioned GMRES, =1 Preconditioned Conjugate gradient. HIPS_DOMSIZE - HIPS_SMOOTH_ITER_RATIO - HIPS developers reserved. HIPS_DOMNBR - HIPS_REORDER - 0 No reordering inside the subdomain to minimize fill-in, =1 reordering inside the subdomain to minimize fill-in. This option is only used in the recursive ITERATIVE preconditioneur. HIPS_SCALENBR - [1-] this value is used to set the number of time the normalisation is applied to the matrix. One should set a value 1 only in special case. HIPS_MASTER - HIPS pretreament (reodering, partitionning...) is done in sequential. The master processor is in charge of these computation. By default it is the processor 0 but you can change this with this option. HIPS_COARSE_GRID - HIPS developers reserved. HIPS_CHECK_GRAPH - [0, 1] (default 1) : set this option to 1 if you want to check (and repair) the matrix adjacency graph. This option ensures the graph is symmetric and that there is no double edge. HIPS_CHECK_MATRIX - =[0, 1] (default 1) : set this option to 1 if you want to check (and repair) the coefficients matrix. This option check if there are coefficient with the same indices and sum them up in this case. HIPS_DUMP_CSR - HIPS developers reserved. HIPS_IMPROVE_PARTITION - HIPS developers reserved. HIPS_TAGNBR - HIPS developers reserved. HIPS_SHIFT_DIAG - HIPS developers reserved. HIPS_GRAPH_SYM - =[0, 1] (default 1) : set this option to 0, if you are sure that the graph you give to HIPS is symmetric ; this disable the graph symmetrization.

HIPS_GRID_DIM - HIPS developers reserved. HIPS_GRID_3D - HIPS developers reserved. HIPS_DISABLE_PRECOND - [0, 1] if set to 1 then when new matrix coefficient are entered the preconditioner is not recalculated in HIPS. Nevertheless, this option is taken into account only if a preconditioner has already been computed.

HIPS_FORTRAN_NUMBERING - [0, 1] (default 1) : numbering in indexes array will start at 0 or 1. This options modify the default numbering for the inputs and returns in all HIPS's functions. HIPS_DOF - [1-] (default 1) : number of unknowns per node in the matrix non-zero pattern graph. Usually, one needs this function when a node represents several degree of freedom. That is to say : any entry (i, j) of the graph represents a dense block of (dof,dof) non null coefficients in the matrix. HIPS_PIVOTING -[0, 1] (default 0) : disable or enable column pivoting in ILUT (only for unsymmetric matrix). This option is not yet fully implemented in parallel.

Enumerator:

HIPS_SYMMETRIC

HIPS_VERBOSE

HIPS_SCALE

HIPS_LOCALLY

HIPS_KRYLOV_RESTART

HIPS_ITMAX

HIPS_FORWARD

HIPS_SCHUR_METHOD

HIPS_ITMAX_SCHUR

HIPS_PARTITION_TYPE

HIPS_KRYLOV_METHOD

HIPS_DOMSIZE

HIPS_SMOOTH_ITER_RATIO

HIPS_DOMNBR

HIPS_REORDER

HIPS_SCALENBR

HIPS_MASTER

HIPS_COARSE_GRID

HIPS_CHECK_GRAPH

HIPS_CHECK_MATRIX

HIPS_DUMP_CSR

HIPS_IMPROVE_PARTITION

HIPS_TAGNBR

HIPS_SHIFT_DIAG

HIPS_GRAPH_SYM

HIPS_GRID_DIM

HIPS_GRID_3D

HIPS_DISABLE_PRECOND
HIPS_FORTRAN_NUMBERING
HIPS_DOF
HIPS_PIVOTING

4.10.1.6 enum HIPS_RETURNS

Enum: HIPS_RETURNS

HIPS error identifiers. These are the list of possible errors returned by HIPS functions. You can call HIPS_PrintError to print the information on a specific error number. You can call HIPS_ExitOnError to print the information on a specific error number and stop the program.

Contains: HIPS_ERR_ALLOCATE , HIPS_ERR_IO , HIPS_ERR_PARAMETER, HIPS_ERR_MATASSEMB, HIPS_ERR_RHSASSEMB, HIPS_ERR_PARASETUP, HIPS_ERR_CALL, HIPS_ERR_PRECOND, HIPS_ERR_SOLVE, HIPS_ERR_-KRYLOV, HIPS_ERR_CHECK,

Enumerator:

HIPS_SUCCESS
HIPS_ERR_ALLOCATE
HIPS_ERR_IO
HIPS_ERR_PARAMETER
HIPS_ERR_MATASSEMB
HIPS_ERR_RHSASSEMB
HIPS_ERR_PARASETUP
HIPS_ERR_CALL
HIPS_ERR_PRECOND
HIPS_ERR_SOLVE
HIPS_ERR_KRYLOV
HIPS_ERR_CHECK

4.10.1.7 enum HIPS_RPARAM

Enum: HIPS_RPARAM

Hips real parameters identifiers.

Solvers may implement is own list of parameters.

Contains: HIPS_PREC - Wanted norm error at the end of solve. HIPS_DROPTOL0 - HIPS_DROPTOL1 - HIPS_DROPTOLSCHUR - HIPS_DROPTOLE - HIPS_-AMALG -

Enumerator:

HIPS_PREC

HIPS_DROPTOL0

HIPS_DROPTOL1

HIPS_DROPSCHUR

HIPS_DROPTOLE

HIPS_AMALG

4.10.1.8 enum HIPS_STRATNUM

Enum: HIPS_STRATNUM

Hips strategy identifiers.

Solvers may implement is own list of parameters.

Contains: HIPS_ITERATIVE - Use the multistage ILUT precondionner HIPS_-HYBRID - Use the hybrid direct/iterative solver

Enumerator:

HIPS_DIRECT

HIPS_ILUT

HIPS_ITERATIVE

HIPS_HYBRID

HIPS_BLOCK

5 Frequently Asked Questions

- **Can I use my own data partition in HIPS ?**
  You can find an example in `TESTS/PARALLEL : testHIPS3.c` or `testHIPS3-Fortran.f90`.
- **How to change the matrix and solve the system without recomputing the preconditioner ?**
  Set the parameter `HIPS_DISABLE_PRECOND` to 0 with the function `HIPS_SetOptionINT`: this allows you to keep the preconditioner unchanged even if you change the matrix.
- **How to use DOF > 1 (degree of freedom) ?**
  Set the parameter `HIPS_DOF` (degree of freedom) to a value > 1 with the function `HIPS_SetOptionINT`.

- **How to save the preprocessing step ?**
  You can find an example in `TESTS/PARALLEL` : `testHIPS-Save.c` and `testHIPS-Load.c`
- **What is the best graph partitioner library to link with ?** (METIS **or** SCOTCH)
  Well, the main SCOTCH developper is in the next office so let say SCOTCH.
- **What is the difference between `nnzP` and `peak` ?**
  `nnzP` is the size, in term of non-zero of the preconditioner but HIPS needs temporary additional memory during the build of the preconditioner. `peak` is the memory peak reach during this step. Using numerical threshold (parameter #10# in Input) you can in general reduce a lot this peak.
- **Why the HIPS logo is an orange spot ?**
  That is a good question.