hypre

2.18.2

Generated by Doxygen 1.9.1

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 HYPRE_IJ_mv.h File Reference

IJ Matrices

- typedef struct hypre_IJMatrix_struct ∗ HYPRE_IJMatrix
- HYPRE_Int HYPRE_IJMatrixCreate (MPI_Comm comm, HYPRE_BigInt ilower, HYPRE_BigInt iupper, HYPRE←_BigInt jlower, HYPRE_BigInt jupper, HYPRE_IJMatrix ∗matrix)
- HYPRE_Int HYPRE_IJMatrixDestroy (HYPRE_IJMatrix matrix)
- HYPRE_Int HYPRE_IJMatrixInitialize (HYPRE_IJMatrix matrix)
- HYPRE_Int HYPRE_IJMatrixSetValues (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_Int ∗ncols, const HYPRE_BigInt ∗rows, const HYPRE_BigInt ∗cols, const HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJMatrixSetConstantValues (HYPRE_IJMatrix matrix, HYPRE_Complex value)
- HYPRE_Int HYPRE_IJMatrixAddToValues (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_Int ∗ncols, const HYPRE_BigInt ∗rows, const HYPRE_BigInt ∗cols, const HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJMatrixSetValues2 (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_Int ∗ncols, const HYPRE_BigInt ∗rows, const HYPRE_Int ∗row_indexes, const HYPRE_BigInt ∗cols, const HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJMatrixAddToValues2 (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_Int ∗ncols, const HYPRE_BigInt ∗rows, const HYPRE_Int ∗row_indexes, const HYPRE_BigInt ∗cols, const HYPRE_←Complex ∗values)
- HYPRE_Int HYPRE_IJMatrixAssemble (HYPRE_IJMatrix matrix)
- HYPRE_Int HYPRE_IJMatrixGetRowCounts (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_BigInt ∗rows, HYPRE_Int ∗ncols)
- HYPRE_Int HYPRE_IJMatrixGetValues (HYPRE_IJMatrix matrix, HYPRE_Int nrows, HYPRE_Int ∗ncols, HYPRE_BigInt ∗rows, HYPRE_BigInt ∗cols, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJMatrixSetObjectType (HYPRE_IJMatrix matrix, HYPRE_Int type)
- HYPRE_Int HYPRE_IJMatrixGetObjectType (HYPRE_IJMatrix matrix, HYPRE_Int ∗type)
- HYPRE_Int HYPRE_IJMatrixGetLocalRange (HYPRE_IJMatrix matrix, HYPRE_BigInt ∗ilower, HYPRE_BigInt ∗iupper, HYPRE_BigInt ∗jlower, HYPRE_BigInt ∗jupper)
- HYPRE_Int HYPRE_IJMatrixGetObject (HYPRE_IJMatrix matrix, void ∗∗object)
- HYPRE_Int HYPRE_IJMatrixSetRowSizes (HYPRE_IJMatrix matrix, const HYPRE_Int ∗sizes)
- HYPRE_Int HYPRE_IJMatrixSetDiagOffdSizes (HYPRE_IJMatrix matrix, const HYPRE_Int ∗diag_sizes, const HYPRE_Int ∗offdiag_sizes)
- HYPRE_Int HYPRE_IJMatrixSetMaxOffProcElmts (HYPRE_IJMatrix matrix, HYPRE_Int max_off_proc_elmts)

- HYPRE_Int HYPRE_IJMatrixSetPrintLevel (HYPRE_IJMatrix matrix, HYPRE_Int print_level)
- HYPRE_Int HYPRE_IJMatrixSetOMPFlag (HYPRE_IJMatrix matrix, HYPRE_Int omp_flag)
- HYPRE_Int HYPRE_IJMatrixRead (const char ∗filename, MPI_Comm comm, HYPRE_Int type, HYPRE_IJMatrix ∗matrix)
- HYPRE_Int HYPRE_IJMatrixPrint (HYPRE_IJMatrix matrix, const char ∗filename)

IJ Vectors

- typedef struct hypre_IJVector_struct ∗ HYPRE_IJVector
- HYPRE_Int HYPRE_IJVectorCreate (MPI_Comm comm, HYPRE_BigInt jlower, HYPRE_BigInt jupper, HYPRE_IJVector ∗vector)
- HYPRE_Int HYPRE_IJVectorDestroy (HYPRE_IJVector vector)
- HYPRE_Int HYPRE_IJVectorInitialize (HYPRE_IJVector vector)
- HYPRE_Int HYPRE_IJVectorSetMaxOffProcElmts (HYPRE_IJVector vector, HYPRE_Int max_off_proc_elmts)
- HYPRE_Int HYPRE_IJVectorSetValues (HYPRE_IJVector vector, HYPRE_Int nvalues, const HYPRE_BigInt ∗indices, const HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJVectorAddToValues (HYPRE_IJVector vector, HYPRE_Int nvalues, const HYPRE_BigInt ∗indices, const HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJVectorAssemble (HYPRE_IJVector vector)
- HYPRE_Int HYPRE_IJVectorGetValues (HYPRE_IJVector vector, HYPRE_Int nvalues, const HYPRE_BigInt ∗indices, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_IJVectorSetObjectType (HYPRE_IJVector vector, HYPRE_Int type)
- HYPRE_Int HYPRE_IJVectorGetObjectType (HYPRE_IJVector vector, HYPRE_Int ∗type)
- HYPRE_Int HYPRE_IJVectorGetLocalRange (HYPRE_IJVector vector, HYPRE_BigInt ∗jlower, HYPRE_BigInt ∗jupper)
- HYPRE_Int HYPRE_IJVectorGetObject (HYPRE_IJVector vector, void ∗∗object)
- HYPRE_Int HYPRE_IJVectorSetPrintLevel (HYPRE_IJVector vector, HYPRE_Int print_level)
- HYPRE_Int HYPRE_IJVectorRead (const char ∗filename, MPI_Comm comm, HYPRE_Int type, HYPRE_IJVector ∗vector)
- HYPRE_Int HYPRE_IJVectorPrint (HYPRE_IJVector vector, const char ∗filename)

2.1.1 Typedef Documentation

2.1.1.1 HYPRE_IJMatrix

```
typedef struct hypre_IJMatrix_struct* HYPRE_IJMatrix
```

The matrix object.

2.1.1.2 HYPRE_IJVector

```
typedef struct hypre_IJVector_struct* HYPRE_IJVector
```

The vector object.

2.1.2 Function Documentation

2.1.2.1 HYPRE_IJMatrixAddToValues()

```
HYPRE_Int HYPRE_IJMatrixAddToValues (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_Int * ncols,
            const HYPRE_BigInt * rows,
            const HYPRE_BigInt * cols,
            const HYPRE_Complex * values )
```

Adds to values for {\tt nrows} rows or partial rows of the matrix.
Usage details are analogous to \Ref{HYPRE_IJMatrixSetValues}.
Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one. Add↩ToValues can be used to add to values on other processors.

Note that a threaded version (threaded over the number of rows) will be called if HYPRE_IJMatrixSetOMPFlag is set to a value != 0. This requires that rows[i] != rows[j] for i!= j and is only efficient if a large number of rows is added in one call to HYPRE_IJMatrixAddToValues.

Not collective.

2.1.2.2 HYPRE_IJMatrixAddToValues2()

```
HYPRE_Int HYPRE_IJMatrixAddToValues2 (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_Int * ncols,
            const HYPRE_BigInt * rows,
            const HYPRE_Int * row_indexes,
            const HYPRE_BigInt * cols,
            const HYPRE_Complex * values )
```

Adds to values for {\tt nrows} rows or partial rows of the matrix.

Same as IJMatrixAddToValues, but with an additional {\tt row_indexes} array that provides indexes into the {\tt cols} and {\tt values} arrays. Because of this, there can be gaps between the row data in these latter two arrays.

2.1.2.3 HYPRE_IJMatrixAssemble()

```
HYPRE_Int HYPRE_IJMatrixAssemble (
            HYPRE_IJMatrix matrix )
```

Finalize the construction of the matrix before using.

2.1.2.4 HYPRE_IJMatrixCreate()

```
HYPRE_Int HYPRE_IJMatrixCreate (
            MPI_Comm comm,
            HYPRE_BigInt ilower,
            HYPRE_BigInt iupper,
            HYPRE_BigInt jlower,
            HYPRE_BigInt jupper,
            HYPRE_IJMatrix * matrix )
```

Create a matrix object. Each process owns some unique consecutive range of rows, indicated by the global row indices {\tt ilower} and {\tt iupper}. The row data is required to be such that the value of {\tt ilower} on any process p be exactly one more than the value of {\tt iupper} on process $p-1$. Note that the first row of the global matrix may start with any integer value. In particular, one may use zero- or one-based indexing.

For square matrices, {\tt jlower} and {\tt jupper} typically should match {\tt ilower} and {\tt iupper}, respectively. For rectangular matrices, {\tt jlower} and {\tt jupper} should define a partitioning of the columns. This partitioning must be used for any vector v that will be used in matrix-vector products with the rectangular matrix. The matrix data structure may use {\tt jlower} and {\tt jupper} to store the diagonal blocks (rectangular in general) of the matrix separately from the rest of the matrix.

Collective.

2.1.2.5 HYPRE_IJMatrixDestroy()

```
HYPRE_Int HYPRE_IJMatrixDestroy (
            HYPRE_IJMatrix matrix )
```

Destroy a matrix object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.1.2.6 HYPRE_IJMatrixGetLocalRange()

```
HYPRE_Int HYPRE_IJMatrixGetLocalRange (
            HYPRE_IJMatrix matrix,
            HYPRE_BigInt * ilower,
            HYPRE_BigInt * iupper,
            HYPRE_BigInt * jlower,
            HYPRE_BigInt * jupper )
```

Gets range of rows owned by this processor and range of column partitioning for this processor.

2.1.2.7 HYPRE_IJMatrixGetObject()

```
HYPRE_Int HYPRE_IJMatrixGetObject (
            HYPRE_IJMatrix matrix,
            void ** object )
```

Get a reference to the constructed matrix object.

See also

HYPRE_IJMatrixSetObjectType

2.1.2.8 HYPRE_IJMatrixGetObjectType()

```
HYPRE_Int HYPRE_IJMatrixGetObjectType (
            HYPRE_IJMatrix matrix,
            HYPRE_Int * type )
```

Get the storage type of the constructed matrix object.

2.1.2.9 HYPRE_IJMatrixGetRowCounts()

```
HYPRE_Int HYPRE_IJMatrixGetRowCounts (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_BigInt * rows,
            HYPRE_Int * ncols )
```

Gets number of nonzeros elements for {\tt nrows} rows specified in {\tt rows} and returns them in {\tt ncols}, which needs to be allocated by the user.

2.1.2.10 HYPRE_IJMatrixGetValues()

```
HYPRE_Int HYPRE_IJMatrixGetValues (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_Int * ncols,
            HYPRE_BigInt * rows,
            HYPRE_BigInt * cols,
            HYPRE_Complex * values )
```

Gets values for {\tt nrows} rows or partial rows of the matrix.
Usage details are mostly analogous to \Ref{HYPRE_IJMatrixSetValues}. Note that if nrows is negative, the routine will return the column_indices and matrix coefficients of the (-nrows) rows contained in rows.

2.1.2.11 HYPRE_IJMatrixInitialize()

```
HYPRE_Int HYPRE_IJMatrixInitialize (
            HYPRE_IJMatrix matrix )
```

Prepare a matrix object for setting coefficient values. This routine will also re-initialize an already assembled matrix, allowing users to modify coefficient values.

2.1.2.12 HYPRE_IJMatrixPrint()

```
HYPRE_Int HYPRE_IJMatrixPrint (
            HYPRE_IJMatrix matrix,
            const char * filename )
```

Print the matrix to file. This is mainly for debugging purposes.

2.1.2.13 HYPRE_IJMatrixRead()

```
HYPRE_Int HYPRE_IJMatrixRead (
            const char * filename,
            MPI_Comm comm,
            HYPRE_Int type,
            HYPRE_IJMatrix * matrix )
```

Read the matrix from file. This is mainly for debugging purposes.

2.1.2.14 HYPRE_IJMatrixSetConstantValues()

```
HYPRE_Int HYPRE_IJMatrixSetConstantValues (
            HYPRE_IJMatrix matrix,
            HYPRE_Complex value )
```

Sets all matrix coefficients of an already assembled matrix to {\tt value}

2.1.2.15 HYPRE_IJMatrixSetDiagOffdSizes()

```
HYPRE_Int HYPRE_IJMatrixSetDiagOffdSizes (
            HYPRE_IJMatrix matrix,
            const HYPRE_Int * diag_sizes,
            const HYPRE_Int * offdiag_sizes )
```

(Optional) Sets the exact number of nonzeros in each row of the diagonal and off-diagonal blocks. The diagonal block is the submatrix whose column numbers correspond to rows owned by this process, and the off-diagonal block is everything else. The arrays {\tt diag_sizes} and {\tt offdiag_sizes} contain estimated sizes for each row of the diagonal and off-diagonal blocks, respectively. This routine can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

2.1.2.16 HYPRE_IJMatrixSetMaxOffProcElmts()

```
HYPRE_Int HYPRE_IJMatrixSetMaxOffProcElmts (
            HYPRE_IJMatrix matrix,
            HYPRE_Int max_off_proc_elmts )
```

(Optional) Sets the maximum number of elements that are expected to be set (or added) on other processors from this processor This routine can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

2.1.2.17 HYPRE_IJMatrixSetObjectType()

```
HYPRE_Int HYPRE_IJMatrixSetObjectType (
            HYPRE_IJMatrix matrix,
            HYPRE_Int type )
```

Set the storage type of the matrix object to be constructed. Currently, {\tt type} can only be {\tt HYPRE_PARCSR}.

Not collective, but must be the same on all processes.

See also

HYPRE_IJMatrixGetObject

2.1.2.18 HYPRE_IJMatrixSetOMPFlag()

```
HYPRE_Int HYPRE_IJMatrixSetOMPFlag (
            HYPRE_IJMatrix matrix,
            HYPRE_Int omp_flag )
```

(Optional) if set, will use a threaded version of HYPRE_IJMatrixSetValues and HYPRE_IJMatrixAddToValues. This is only useful if a large number of rows is set or added to at once.

NOTE that the values in the rows array of HYPRE_IJMatrixSetValues or HYPRE_IJMatrixAddToValues must be different from each other !!!

This option is VERY inefficient if only a small number of rows is set or added at once and/or if reallocation of storage is required and/or if values are added to off processor values.

2.1.2.19 HYPRE_IJMatrixSetPrintLevel()

```
HYPRE_Int HYPRE_IJMatrixSetPrintLevel (
            HYPRE_IJMatrix matrix,
            HYPRE_Int print_level )
```

(Optional) Sets the print level, if the user wants to print error messages. The default is 0, i.e. no error messages are printed.

2.1.2.20 HYPRE_IJMatrixSetRowSizes()

```
HYPRE_Int HYPRE_IJMatrixSetRowSizes (
            HYPRE_IJMatrix matrix,
            const HYPRE_Int * sizes )
```

(Optional) Set the max number of nonzeros to expect in each row. The array {\tt sizes} contains estimated sizes for each row on this process. This call can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

2.1.2.21 HYPRE_IJMatrixSetValues()

```
HYPRE_Int HYPRE_IJMatrixSetValues (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_Int * ncols,
            const HYPRE_BigInt * rows,
            const HYPRE_BigInt * cols,
            const HYPRE_Complex * values )
```

Sets values for {\tt nrows} rows or partial rows of the matrix.
The arrays {\tt ncols} and {\tt rows} are of dimension {\tt nrows} and contain the number of columns in each row and the row indices, respectively. The array {\tt cols} contains the column indices for each of the {\tt rows}, and is ordered by rows. The data in the {\tt values} array corresponds directly to the column entries in {\tt cols}. Erases any previous values at the specified locations and replaces them with new ones, or, if there was no value there before, inserts a new one if set locally. Note that it is not possible to set values on other processors. If one tries to set a value from proc i on proc j, proc i will erase all previous occurrences of this value in its stack (including values generated with AddToValues), and treat it like a zero value. The actual value needs to be set on proc j.

Note that a threaded version (threaded over the number of rows) will be called if HYPRE_IJMatrixSetOMPFlag is set to a value != 0. This requires that rows[i] != rows[j] for i!= j and is only efficient if a large number of rows is set in one call to HYPRE_IJMatrixSetValues.

Not collective.

2.1.2.22 HYPRE_IJMatrixSetValues2()

```
HYPRE_Int HYPRE_IJMatrixSetValues2 (
            HYPRE_IJMatrix matrix,
            HYPRE_Int nrows,
            HYPRE_Int * ncols,
            const HYPRE_BigInt * rows,
            const HYPRE_Int * row_indexes,
            const HYPRE_BigInt * cols,
            const HYPRE_Complex * values )
```

Sets values for {\tt nrows} rows or partial rows of the matrix.

Same as IJMatrixSetValues, but with an additional {\tt row_indexes} array that provides indexes into the {\tt cols} and {\tt values} arrays. Because of this, there can be gaps between the row data in these latter two arrays.

2.1.2.23 HYPRE_IJVectorAddToValues()

```
HYPRE_Int HYPRE_IJVectorAddToValues (
            HYPRE_IJVector vector,
            HYPRE_Int nvalues,
            const HYPRE_BigInt * indices,
            const HYPRE_Complex * values )
```

Adds to values in vector. Usage details are analogous to \Ref{HYPRE_IJVectorSetValues}. Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one. AddToValues can be used to add to values on other processors.

Not collective.

2.1.2.24 HYPRE_IJVectorAssemble()

```
HYPRE_Int HYPRE_IJVectorAssemble (
            HYPRE_IJVector vector )
```

Finalize the construction of the vector before using.

2.1.2.25 HYPRE_IJVectorCreate()

```
HYPRE_Int HYPRE_IJVectorCreate (
            MPI_Comm comm,
            HYPRE_BigInt jlower,
            HYPRE_BigInt jupper,
            HYPRE_IJVector * vector )
```

Create a vector object. Each process owns some unique consecutive range of vector unknowns, indicated by the global indices {\tt jlower} and {\tt jupper}. The data is required to be such that the value of {\tt jlower} on any process p be exactly one more than the value of {\tt jupper} on process $p-1$. Note that the first index of the global vector may start with any integer value. In particular, one may use zero- or one-based indexing.

Collective.

2.1.2.26 HYPRE_IJVectorDestroy()

```
HYPRE_Int HYPRE_IJVectorDestroy (
            HYPRE_IJVector vector )
```

Destroy a vector object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.1.2.27 HYPRE_IJVectorGetLocalRange()

```
HYPRE_Int HYPRE_IJVectorGetLocalRange (
            HYPRE_IJVector vector,
            HYPRE_BigInt * jlower,
            HYPRE_BigInt * jupper )
```

Returns range of the part of the vector owned by this processor.

2.1.2.28 HYPRE_IJVectorGetObject()

```
HYPRE_Int HYPRE_IJVectorGetObject (
            HYPRE_IJVector vector,
            void ** object )
```

Get a reference to the constructed vector object.

See also

HYPRE_IJVectorSetObjectType

2.1.2.29 HYPRE_IJVectorGetObjectType()

```
HYPRE_Int HYPRE_IJVectorGetObjectType (
            HYPRE_IJVector vector,
            HYPRE_Int * type )
```

Get the storage type of the constructed vector object.

2.1.2.30 HYPRE_IJVectorGetValues()

```
HYPRE_Int HYPRE_IJVectorGetValues (
            HYPRE_IJVector vector,
            HYPRE_Int nvalues,
            const HYPRE_BigInt * indices,
            HYPRE_Complex * values )
```

Gets values in vector. Usage details are analogous to \Ref{HYPRE_IJVectorSetValues}.

Not collective.

2.1.2.31 HYPRE_IJVectorInitialize()

```
HYPRE_Int HYPRE_IJVectorInitialize (
            HYPRE_IJVector vector )
```

Prepare a vector object for setting coefficient values. This routine will also re-initialize an already assembled vector, allowing users to modify coefficient values.

2.1.2.32 HYPRE_IJVectorPrint()

```
HYPRE_Int HYPRE_IJVectorPrint (
            HYPRE_IJVector vector,
            const char * filename )
```

Print the vector to file. This is mainly for debugging purposes.

2.1.2.33 HYPRE_IJVectorRead()

```
HYPRE_Int HYPRE_IJVectorRead (
            const char * filename,
            MPI_Comm comm,
            HYPRE_Int type,
            HYPRE_IJVector * vector )
```

Read the vector from file. This is mainly for debugging purposes.

2.1.2.34 HYPRE_IJVectorSetMaxOffProcElmts()

```
HYPRE_Int HYPRE_IJVectorSetMaxOffProcElmts (
            HYPRE_IJVector vector,
            HYPRE_Int max_off_proc_elmts )
```

(Optional) Sets the maximum number of elements that are expected to be set (or added) on other processors from this processor This routine can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

2.1.2.35 HYPRE_IJVectorSetObjectType()

```
HYPRE_Int HYPRE_IJVectorSetObjectType (
            HYPRE_IJVector vector,
            HYPRE_Int type )
```

Set the storage type of the vector object to be constructed. Currently, {\tt type} can only be {\tt HYPRE_PARCSR}.

Not collective, but must be the same on all processes.

See also

HYPRE_IJVectorGetObject

2.1.2.36 HYPRE_IJVectorSetPrintLevel()

```
HYPRE_Int HYPRE_IJVectorSetPrintLevel (
            HYPRE_IJVector vector,
            HYPRE_Int print_level )
```

(Optional) Sets the print level, if the user wants to print error messages. The default is 0, i.e. no error messages are printed.

2.1.2.37 HYPRE_IJVectorSetValues()

```
HYPRE_Int HYPRE_IJVectorSetValues (
            HYPRE_IJVector vector,
            HYPRE_Int nvalues,
            const HYPRE_BigInt * indices,
            const HYPRE_Complex * values )
```

Sets values in vector. The arrays {\tt values} and {\tt indices} are of dimension {\tt nvalues} and contain the vector values to be set and the corresponding global vector indices, respectively. Erases any previous values at the specified locations and replaces them with new ones. Note that it is not possible to set values on other processors. If one tries to set a value from proc i on proc j, proc i will erase all previous occurrences of this value in its stack (including values generated with AddToValues), and treat it like a zero value. The actual value needs to be set on proc j.

Not collective.

2.2 HYPRE_krylov.h File Reference

Functions

PCG Solver

- HYPRE_Int HYPRE_PCGSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_PCGSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_PCGSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_PCGSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_PCGSetResidualTol (HYPRE_Solver solver, HYPRE_Real rtol)
- HYPRE_Int HYPRE_PCGSetAbsoluteTolFactor (HYPRE_Solver solver, HYPRE_Real abstolf)
- HYPRE_Int HYPRE_PCGSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_PCGSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_PCGSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_PCGSetTwoNorm (HYPRE_Solver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_PCGSetRelChange (HYPRE_Solver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_PCGSetRecomputeResidual (HYPRE_Solver solver, HYPRE_Int recompute_residual)
- HYPRE_Int HYPRE_PCGSetRecomputeResidualP (HYPRE_Solver solver, HYPRE_Int recompute_↩ residual_p)
- HYPRE_Int HYPRE_PCGSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_PCGSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_PCGSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)

- HYPRE_Int HYPRE_PCGGetNumIterations (HYPRE_Solver solver, HYPRE_Int *num_iterations)
- HYPRE_Int HYPRE_PCGGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real *norm)
- HYPRE_Int HYPRE_PCGGetResidual (HYPRE_Solver solver, void *residual)
- HYPRE_Int HYPRE_PCGGetTol (HYPRE_Solver solver, HYPRE_Real *tol)
- HYPRE_Int HYPRE_PCGGetResidualTol (HYPRE_Solver solver, HYPRE_Real *rtol)
- HYPRE_Int HYPRE_PCGGetAbsoluteTolFactor (HYPRE_Solver solver, HYPRE_Real *abstolf)
- HYPRE_Int HYPRE_PCGGetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real *cf_tol)
- HYPRE_Int HYPRE_PCGGetStopCrit (HYPRE_Solver solver, HYPRE_Int *stop_crit)
- HYPRE_Int HYPRE_PCGGetMaxIter (HYPRE_Solver solver, HYPRE_Int *max_iter)
- HYPRE_Int HYPRE_PCGGetTwoNorm (HYPRE_Solver solver, HYPRE_Int *two_norm)
- HYPRE_Int HYPRE_PCGGetRelChange (HYPRE_Solver solver, HYPRE_Int *rel_change)
- HYPRE_Int HYPRE_GMRESGetSkipRealResidualCheck (HYPRE_Solver solver, HYPRE_Int *skip_real_r←_check)
- HYPRE_Int HYPRE_PCGGetPrecond (HYPRE_Solver solver, HYPRE_Solver *precond_data_ptr)
- HYPRE_Int HYPRE_PCGGetLogging (HYPRE_Solver solver, HYPRE_Int *level)
- HYPRE_Int HYPRE_PCGGetPrintLevel (HYPRE_Solver solver, HYPRE_Int *level)
- HYPRE_Int HYPRE_PCGGetConverged (HYPRE_Solver solver, HYPRE_Int *converged)

GMRES Solver

- HYPRE_Int HYPRE_GMRESSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_GMRESSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_GMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_GMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_GMRESSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_GMRESSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_GMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_GMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_GMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_GMRESSetRelChange (HYPRE_Solver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_GMRESSetSkipRealResidualCheck (HYPRE_Solver solver, HYPRE_Int skip_real_r_←check)
- HYPRE_Int HYPRE_GMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_GMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_GMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_GMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int *num_iterations)
- HYPRE_Int HYPRE_GMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real *norm)
- HYPRE_Int HYPRE_GMRESGetResidual (HYPRE_Solver solver, void *residual)
- HYPRE_Int HYPRE_GMRESGetTol (HYPRE_Solver solver, HYPRE_Real *tol)
- HYPRE_Int HYPRE_GMRESGetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real *tol)
- HYPRE_Int HYPRE_GMRESGetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real *cf_tol)
- HYPRE_Int HYPRE_GMRESGetStopCrit (HYPRE_Solver solver, HYPRE_Int *stop_crit)
- HYPRE_Int HYPRE_GMRESGetMinIter (HYPRE_Solver solver, HYPRE_Int *min_iter)
- HYPRE_Int HYPRE_GMRESGetMaxIter (HYPRE_Solver solver, HYPRE_Int *max_iter)
- HYPRE_Int HYPRE_GMRESGetKDim (HYPRE_Solver solver, HYPRE_Int *k_dim)
- HYPRE_Int HYPRE_GMRESGetRelChange (HYPRE_Solver solver, HYPRE_Int *rel_change)
- HYPRE_Int HYPRE_GMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver *precond_data_ptr)
- HYPRE_Int HYPRE_GMRESGetLogging (HYPRE_Solver solver, HYPRE_Int *level)
- HYPRE_Int HYPRE_GMRESGetPrintLevel (HYPRE_Solver solver, HYPRE_Int *level)
- HYPRE_Int HYPRE_GMRESGetConverged (HYPRE_Solver solver, HYPRE_Int *converged)

FlexGMRES Solver

- HYPRE_Int HYPRE_FlexGMRESSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)

- HYPRE_Int HYPRE_FlexGMRESSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_FlexGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_FlexGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_FlexGMRESSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_FlexGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_FlexGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_FlexGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_FlexGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_FlexGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_FlexGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_FlexGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_FlexGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_FlexGMRESGetResidual (HYPRE_Solver solver, void ∗residual)
- HYPRE_Int HYPRE_FlexGMRESGetTol (HYPRE_Solver solver, HYPRE_Real ∗tol)
- HYPRE_Int HYPRE_FlexGMRESGetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real ∗cf_tol)
- HYPRE_Int HYPRE_FlexGMRESGetStopCrit (HYPRE_Solver solver, HYPRE_Int ∗stop_crit)
- HYPRE_Int HYPRE_FlexGMRESGetMinIter (HYPRE_Solver solver, HYPRE_Int ∗min_iter)
- HYPRE_Int HYPRE_FlexGMRESGetMaxIter (HYPRE_Solver solver, HYPRE_Int ∗max_iter)
- HYPRE_Int HYPRE_FlexGMRESGetKDim (HYPRE_Solver solver, HYPRE_Int ∗k_dim)
- HYPRE_Int HYPRE_FlexGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data_ptr)
- HYPRE_Int HYPRE_FlexGMRESGetLogging (HYPRE_Solver solver, HYPRE_Int ∗level)
- HYPRE_Int HYPRE_FlexGMRESGetPrintLevel (HYPRE_Solver solver, HYPRE_Int ∗level)
- HYPRE_Int HYPRE_FlexGMRESGetConverged (HYPRE_Solver solver, HYPRE_Int ∗converged)
- HYPRE_Int HYPRE_FlexGMRESSetModifyPC (HYPRE_Solver solver, HYPRE_PtrToModifyPCFcn modify↩ _pc)

LGMRES Solver

- HYPRE_Int HYPRE_LGMRESSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_LGMRESSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_LGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_LGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_LGMRESSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_LGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_LGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_LGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_LGMRESSetAugDim (HYPRE_Solver solver, HYPRE_Int aug_dim)
- HYPRE_Int HYPRE_LGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_LGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_LGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_LGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_LGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_LGMRESGetResidual (HYPRE_Solver solver, void ∗residual)
- HYPRE_Int HYPRE_LGMRESGetTol (HYPRE_Solver solver, HYPRE_Real ∗tol)
- HYPRE_Int HYPRE_LGMRESGetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real ∗cf_tol)
- HYPRE_Int HYPRE_LGMRESGetStopCrit (HYPRE_Solver solver, HYPRE_Int ∗stop_crit)
- HYPRE_Int HYPRE_LGMRESGetMinIter (HYPRE_Solver solver, HYPRE_Int ∗min_iter)
- HYPRE_Int HYPRE_LGMRESGetMaxIter (HYPRE_Solver solver, HYPRE_Int ∗max_iter)
- HYPRE_Int HYPRE_LGMRESGetKDim (HYPRE_Solver solver, HYPRE_Int ∗k_dim)
- HYPRE_Int HYPRE_LGMRESGetAugDim (HYPRE_Solver solver, HYPRE_Int ∗k_dim)
- HYPRE_Int HYPRE_LGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data_ptr)
- HYPRE_Int HYPRE_LGMRESGetLogging (HYPRE_Solver solver, HYPRE_Int ∗level)

- HYPRE_Int HYPRE_LGMRESGetPrintLevel (HYPRE_Solver solver, HYPRE_Int ∗level)
- HYPRE_Int HYPRE_LGMRESGetConverged (HYPRE_Solver solver, HYPRE_Int ∗converged)

COGMRES Solver

- HYPRE_Int HYPRE_COGMRESSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_COGMRESSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_COGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_COGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_COGMRESSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_COGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_COGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_COGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_COGMRESSetUnroll (HYPRE_Solver solver, HYPRE_Int unroll)
- HYPRE_Int HYPRE_COGMRESSetCGS (HYPRE_Solver solver, HYPRE_Int cgs)
- HYPRE_Int HYPRE_COGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_COGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_COGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_COGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_COGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_COGMRESGetResidual (HYPRE_Solver solver, void ∗residual)
- HYPRE_Int HYPRE_COGMRESGetTol (HYPRE_Solver solver, HYPRE_Real ∗tol)
- HYPRE_Int HYPRE_COGMRESGetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real ∗cf_tol)
- HYPRE_Int HYPRE_COGMRESGetMinIter (HYPRE_Solver solver, HYPRE_Int ∗min_iter)
- HYPRE_Int HYPRE_COGMRESGetMaxIter (HYPRE_Solver solver, HYPRE_Int ∗max_iter)
- HYPRE_Int HYPRE_COGMRESGetKDim (HYPRE_Solver solver, HYPRE_Int ∗k_dim)
- HYPRE_Int HYPRE_COGMRESGetUnroll (HYPRE_Solver solver, HYPRE_Int ∗unroll)
- HYPRE_Int HYPRE_COGMRESGetCGS (HYPRE_Solver solver, HYPRE_Int ∗cgs)
- HYPRE_Int HYPRE_COGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data_ptr)
- HYPRE_Int HYPRE_COGMRESGetLogging (HYPRE_Solver solver, HYPRE_Int ∗level)
- HYPRE_Int HYPRE_COGMRESGetPrintLevel (HYPRE_Solver solver, HYPRE_Int ∗level)
- HYPRE_Int HYPRE_COGMRESGetConverged (HYPRE_Solver solver, HYPRE_Int ∗converged)
- HYPRE_Int HYPRE_COGMRESSetModifyPC (HYPRE_Solver solver, HYPRE_PtrToModifyPCFcn modify_pc)

BiCGSTAB Solver

- HYPRE_Int HYPRE_BiCGSTABDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_BiCGSTABSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_BiCGSTABSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_BiCGSTABSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_BiCGSTABSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_BiCGSTABSetConvergenceFactorTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_BiCGSTABSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_BiCGSTABSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_BiCGSTABSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_BiCGSTABSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_BiCGSTABSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_BiCGSTABSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_BiCGSTABGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_BiCGSTABGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)

- HYPRE_Int HYPRE_BiCGSTABGetResidual (HYPRE_Solver solver, void ∗residual)
- HYPRE_Int HYPRE_BiCGSTABGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data_ptr)

CGNR Solver

- HYPRE_Int HYPRE_CGNRDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_CGNRSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_CGNRSolve (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_CGNRSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_CGNRSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_CGNRSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_CGNRSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_CGNRSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precondT, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_CGNRSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_CGNRGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_CGNRGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_CGNRGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data_ptr)

Krylov Solvers

- #define HYPRE_SOLVER_STRUCT
- #define HYPRE_MATRIX_STRUCT
- #define HYPRE_VECTOR_STRUCT
- #define HYPRE_MODIFYPC
- typedef struct hypre_Solver_struct ∗ HYPRE_Solver
- typedef struct hypre_Matrix_struct ∗ HYPRE_Matrix
- typedef struct hypre_Vector_struct ∗ HYPRE_Vector
- typedef HYPRE_Int(∗ HYPRE_PtrToSolverFcn) (HYPRE_Solver, HYPRE_Matrix, HYPRE_Vector, HYPRE_Vector)
- typedef HYPRE_Int(∗ HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)

2.2.1 Macro Definition Documentation

2.2.1.1 HYPRE_MATRIX_STRUCT

```
#define HYPRE_MATRIX_STRUCT
```

2.2.1.2 HYPRE_MODIFYPC

```
#define HYPRE_MODIFYPC
```

2.2.1.3 HYPRE_SOLVER_STRUCT

```
#define HYPRE_SOLVER_STRUCT
```

2.2.1.4 HYPRE_VECTOR_STRUCT

```
#define HYPRE_VECTOR_STRUCT
```

2.2.2 Typedef Documentation

2.2.2.1 HYPRE_Matrix

```
typedef struct hypre_Matrix_struct* HYPRE_Matrix
```

The matrix object.

2.2.2.2 HYPRE_PtrToModifyPCFcn

```
typedef HYPRE_Int(* HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)
```

2.2.2.3 HYPRE_PtrToSolverFcn

```
typedef HYPRE_Int(* HYPRE_PtrToSolverFcn) (HYPRE_Solver, HYPRE_Matrix, HYPRE_Vector, HYPRE_Vector)
```

2.2.2.4 HYPRE_Solver

```
typedef struct hypre_Solver_struct* HYPRE_Solver
```

The solver object.

2.2.2.5 HYPRE_Vector

```
typedef struct hypre_Vector_struct* HYPRE_Vector
```

The vector object.

2.2.3 Function Documentation

2.2.3.1 HYPRE_BiCGSTABDestroy()

```
HYPRE_Int HYPRE_BiCGSTABDestroy (
            HYPRE_Solver solver )
```

2.2.3.2 HYPRE_BiCGSTABGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_BiCGSTABGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.3 HYPRE_BiCGSTABGetNumIterations()

```
HYPRE_Int HYPRE_BiCGSTABGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.4 HYPRE_BiCGSTABGetPrecond()

```
HYPRE_Int HYPRE_BiCGSTABGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.5 HYPRE_BiCGSTABGetResidual()

```
HYPRE_Int HYPRE_BiCGSTABGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.6 HYPRE_BiCGSTABSetAbsoluteTol()

```
HYPRE_Int HYPRE_BiCGSTABSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The convergence test is $|r| \leq$ max(relative$_$tolerance $\ast |b|$, absolute$_$tolerance).)

2.2.3.7 HYPRE_BiCGSTABSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_BiCGSTABSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.8 HYPRE_BiCGSTABSetLogging()

```
HYPRE_Int HYPRE_BiCGSTABSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.9 HYPRE_BiCGSTABSetMaxIter()

```
HYPRE_Int HYPRE_BiCGSTABSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.10 HYPRE_BiCGSTABSetMinIter()

```
HYPRE_Int HYPRE_BiCGSTABSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.11 HYPRE_BiCGSTABSetPrecond()

```
HYPRE_Int HYPRE_BiCGSTABSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.12 HYPRE_BiCGSTABSetPrintLevel()

```
HYPRE_Int HYPRE_BiCGSTABSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.13 HYPRE_BiCGSTABSetStopCrit()

```
HYPRE_Int HYPRE_BiCGSTABSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.2.3.14 HYPRE_BiCGSTABSetTol()

```
HYPRE_Int HYPRE_BiCGSTABSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.2.3.15 HYPRE_BiCGSTABSetup()

```
HYPRE_Int HYPRE_BiCGSTABSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.16 HYPRE_BiCGSTABSolve()

```
HYPRE_Int HYPRE_BiCGSTABSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.2.3.17 HYPRE_CGNRDestroy()

```
HYPRE_Int HYPRE_CGNRDestroy (
            HYPRE_Solver solver )
```

2.2.3.18 HYPRE_CGNRGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_CGNRGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.19 HYPRE_CGNRGetNumIterations()

```
HYPRE_Int HYPRE_CGNRGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.20 HYPRE_CGNRGetPrecond()

```
HYPRE_Int HYPRE_CGNRGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.21 HYPRE_CGNRSetLogging()

```
HYPRE_Int HYPRE_CGNRSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.22 HYPRE_CGNRSetMaxIter()

```
HYPRE_Int HYPRE_CGNRSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.23 HYPRE_CGNRSetMinIter()

```
HYPRE_Int HYPRE_CGNRSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.24 HYPRE_CGNRSetPrecond()

```
HYPRE_Int HYPRE_CGNRSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precondT,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use. Note that the only preconditioner available in hypre for use with CGNR is currently BoomerAMG. It requires to use Jacobi as a smoother without CF smoothing, i.e. relax_type needs to be set to 0 or 7 and relax_order needs to be set to 0 by the user, since these are not default values. It can be used with a relaxation weight for Jacobi, which can significantly improve convergence.

2.2.3.25 HYPRE_CGNRSetStopCrit()

```
HYPRE_Int HYPRE_CGNRSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.2.3.26 HYPRE_CGNRSetTol()

```
HYPRE_Int HYPRE_CGNRSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.2.3.27 HYPRE_CGNRSetup()

```
HYPRE_Int HYPRE_CGNRSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.28 HYPRE_CGNRSolve()

```
HYPRE_Int HYPRE_CGNRSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.2.3.29 HYPRE_COGMRESGetCGS()

```
HYPRE_Int HYPRE_COGMRESGetCGS (
            HYPRE_Solver solver,
            HYPRE_Int * cgs )
```

2.2.3.30 HYPRE_COGMRESGetConverged()

```
HYPRE_Int HYPRE_COGMRESGetConverged (
            HYPRE_Solver solver,
            HYPRE_Int * converged )
```

2.2.3.31 HYPRE_COGMRESGetConvergenceFactorTol()

```
HYPRE_Int HYPRE_COGMRESGetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real * cf_tol )
```

2.2.3.32 HYPRE_COGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_COGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.33 HYPRE_COGMRESGetKDim()

```
HYPRE_Int HYPRE_COGMRESGetKDim (
            HYPRE_Solver solver,
            HYPRE_Int * k_dim )
```

2.2.3.34 HYPRE_COGMRESGetLogging()

```
HYPRE_Int HYPRE_COGMRESGetLogging (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.35 HYPRE_COGMRESGetMaxIter()

```
HYPRE_Int HYPRE_COGMRESGetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int * max_iter )
```

2.2.3.36 HYPRE_COGMRESGetMinIter()

```
HYPRE_Int HYPRE_COGMRESGetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int * min_iter )
```

2.2.3.37 HYPRE_COGMRESGetNumIterations()

```
HYPRE_Int HYPRE_COGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.38 HYPRE_COGMRESGetPrecond()

```
HYPRE_Int HYPRE_COGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.39 HYPRE_COGMRESGetPrintLevel()

```
HYPRE_Int HYPRE_COGMRESGetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.40 HYPRE_COGMRESGetResidual()

```
HYPRE_Int HYPRE_COGMRESGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.41 HYPRE_COGMRESGetTol()

```
HYPRE_Int HYPRE_COGMRESGetTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.42 HYPRE_COGMRESGetUnroll()

```
HYPRE_Int HYPRE_COGMRESGetUnroll (
            HYPRE_Solver solver,
            HYPRE_Int * unroll )
```

2.2.3.43 HYPRE_COGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_COGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The convergence test is $|r| \leq$ max(relative$_$tolerance$\ast |b|$, absolute$_$tolerance).)

2.2.3.44 HYPRE_COGMRESSetCGS()

```
HYPRE_Int HYPRE_COGMRESSetCGS (
            HYPRE_Solver solver,
            HYPRE_Int cgs )
```

(Optional) Set the number of orthogonalizations in COGMRES (at most 2).

2.2.3.45 HYPRE_COGMRESSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_COGMRESSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.46 HYPRE_COGMRESSetKDim()

```
HYPRE_Int HYPRE_COGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

(Optional) Set the maximum size of the Krylov space.

2.2.3.47 HYPRE_COGMRESSetLogging()

```
HYPRE_Int HYPRE_COGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.48 HYPRE_COGMRESSetMaxIter()

```
HYPRE_Int HYPRE_COGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.49 HYPRE_COGMRESSetMinIter()

```
HYPRE_Int HYPRE_COGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.50 HYPRE_COGMRESSetModifyPC()

```
HYPRE_Int HYPRE_COGMRESSetModifyPC (
            HYPRE_Solver solver,
            HYPRE_PtrToModifyPCFcn modify_pc )
```

(Optional) Set a user-defined function to modify solve-time preconditioner attributes.

2.2.3.51 HYPRE_COGMRESSetPrecond()

```
HYPRE_Int HYPRE_COGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.52 HYPRE_COGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_COGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.53 HYPRE_COGMRESSetTol()

```
HYPRE_Int HYPRE_COGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.2.3.54 HYPRE_COGMRESSetUnroll()

```
HYPRE_Int HYPRE_COGMRESSetUnroll (
            HYPRE_Solver solver,
            HYPRE_Int unroll )
```

(Optional) Set number of unrolling in mass funcyions in COGMRES Can be 4 or 8. Default: no unrolling.

2.2.3.55 HYPRE_COGMRESSetup()

```
HYPRE_Int HYPRE_COGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.56 HYPRE_COGMRESSolve()

```
HYPRE_Int HYPRE_COGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.2.3.57 HYPRE_FlexGMRESGetConverged()

```
HYPRE_Int HYPRE_FlexGMRESGetConverged (
            HYPRE_Solver solver,
            HYPRE_Int * converged )
```

2.2.3.58 HYPRE_FlexGMRESGetConvergenceFactorTol()

```
HYPRE_Int HYPRE_FlexGMRESGetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real * cf_tol )
```

2.2.3.59 HYPRE_FlexGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_FlexGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.60 HYPRE_FlexGMRESGetKDim()

```
HYPRE_Int HYPRE_FlexGMRESGetKDim (
            HYPRE_Solver solver,
            HYPRE_Int * k_dim )
```

2.2.3.61 HYPRE_FlexGMRESGetLogging()

```
HYPRE_Int HYPRE_FlexGMRESGetLogging (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.62 HYPRE_FlexGMRESGetMaxIter()

```
HYPRE_Int HYPRE_FlexGMRESGetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int * max_iter )
```

2.2.3.63 HYPRE_FlexGMRESGetMinIter()

```
HYPRE_Int HYPRE_FlexGMRESGetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int * min_iter )
```

2.2.3.64 HYPRE_FlexGMRESGetNumIterations()

```
HYPRE_Int HYPRE_FlexGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.65 HYPRE_FlexGMRESGetPrecond()

```
HYPRE_Int HYPRE_FlexGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.66 HYPRE_FlexGMRESGetPrintLevel()

```
HYPRE_Int HYPRE_FlexGMRESGetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.67 HYPRE_FlexGMRESGetResidual()

```
HYPRE_Int HYPRE_FlexGMRESGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.68 HYPRE_FlexGMRESGetStopCrit()

```
HYPRE_Int HYPRE_FlexGMRESGetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int * stop_crit )
```

2.2.3.69 HYPRE_FlexGMRESGetTol()

```
HYPRE_Int HYPRE_FlexGMRESGetTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.70 HYPRE_FlexGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_FlexGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The convergence test is $|r| \leq$ max(relative$_$tolerance$\ast |b|$, absolute$_$tolerance).)

2.2.3.71 HYPRE_FlexGMRESSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_FlexGMRESSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.72 HYPRE_FlexGMRESSetKDim()

```
HYPRE_Int HYPRE_FlexGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

(Optional) Set the maximum size of the Krylov space.

2.2.3.73 HYPRE_FlexGMRESSetLogging()

```
HYPRE_Int HYPRE_FlexGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.74 HYPRE_FlexGMRESSetMaxIter()

```
HYPRE_Int HYPRE_FlexGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.75 HYPRE_FlexGMRESSetMinIter()

```
HYPRE_Int HYPRE_FlexGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.76 HYPRE_FlexGMRESSetModifyPC()

```
HYPRE_Int HYPRE_FlexGMRESSetModifyPC (
            HYPRE_Solver solver,
            HYPRE_PtrToModifyPCFcn modify_pc )
```

(Optional) Set a user-defined function to modify solve-time preconditioner attributes.

2.2.3.77 HYPRE_FlexGMRESSetPrecond()

```
HYPRE_Int HYPRE_FlexGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.78 HYPRE_FlexGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_FlexGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.79 HYPRE_FlexGMRESSetTol()

```
HYPRE_Int HYPRE_FlexGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.2.3.80 HYPRE_FlexGMRESSetup()

```
HYPRE_Int HYPRE_FlexGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.81 HYPRE_FlexGMRESSolve()

```
HYPRE_Int HYPRE_FlexGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.2.3.82 HYPRE_GMRESGetAbsoluteTol()

```
HYPRE_Int HYPRE_GMRESGetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.83 HYPRE_GMRESGetConverged()

```
HYPRE_Int HYPRE_GMRESGetConverged (
            HYPRE_Solver solver,
            HYPRE_Int * converged )
```

2.2.3.84 HYPRE_GMRESGetConvergenceFactorTol()

```
HYPRE_Int HYPRE_GMRESGetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real * cf_tol )
```

2.2.3.85 HYPRE_GMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_GMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.86 HYPRE_GMRESGetKDim()

```
HYPRE_Int HYPRE_GMRESGetKDim (
            HYPRE_Solver solver,
            HYPRE_Int * k_dim )
```

2.2.3.87 HYPRE_GMRESGetLogging()

```
HYPRE_Int HYPRE_GMRESGetLogging (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.88 HYPRE_GMRESGetMaxIter()

```
HYPRE_Int HYPRE_GMRESGetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int * max_iter )
```

2.2.3.89 HYPRE_GMRESGetMinIter()

```
HYPRE_Int HYPRE_GMRESGetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int * min_iter )
```

2.2.3.90 HYPRE_GMRESGetNumIterations()

```
HYPRE_Int HYPRE_GMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.91 HYPRE_GMRESGetPrecond()

```
HYPRE_Int HYPRE_GMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.92 HYPRE_GMRESGetPrintLevel()

```
HYPRE_Int HYPRE_GMRESGetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.93 HYPRE_GMRESGetRelChange()

```
HYPRE_Int HYPRE_GMRESGetRelChange (
            HYPRE_Solver solver,
            HYPRE_Int * rel_change )
```

2.2.3.94 HYPRE_GMRESGetResidual()

```
HYPRE_Int HYPRE_GMRESGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.95 HYPRE_GMRESGetSkipRealResidualCheck()

```
HYPRE_Int HYPRE_GMRESGetSkipRealResidualCheck (
            HYPRE_Solver solver,
            HYPRE_Int * skip_real_r_check )
```

2.2.3.96 HYPRE_GMRESGetStopCrit()

```
HYPRE_Int HYPRE_GMRESGetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int * stop_crit )
```

2.2.3.97 HYPRE_GMRESGetTol()

```
HYPRE_Int HYPRE_GMRESGetTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.98 HYPRE_GMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_GMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The convergence test is $|r| \leq$ max(relative$_$tolerance$\ast |b|$, absolute$_$tolerance).)

2.2.3.99 HYPRE_GMRESSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_GMRESSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.100 HYPRE_GMRESSetKDim()

```
HYPRE_Int HYPRE_GMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

(Optional) Set the maximum size of the Krylov space.

2.2.3.101 HYPRE_GMRESSetLogging()

```
HYPRE_Int HYPRE_GMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.102 HYPRE_GMRESSetMaxIter()

```
HYPRE_Int HYPRE_GMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.103 HYPRE_GMRESSetMinIter()

```
HYPRE_Int HYPRE_GMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.104 HYPRE_GMRESSetPrecond()

```
HYPRE_Int HYPRE_GMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.105 HYPRE_GMRESSetPrintLevel()

```
HYPRE_Int HYPRE_GMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.106 HYPRE_GMRESSetRelChange()

```
HYPRE_Int HYPRE_GMRESSetRelChange (
            HYPRE_Solver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.2.3.107 HYPRE_GMRESSetSkipRealResidualCheck()

```
HYPRE_Int HYPRE_GMRESSetSkipRealResidualCheck (
            HYPRE_Solver solver,
            HYPRE_Int skip_real_r_check )
```

(Optional) By default, hypre checks for convergence by evaluating the actual residual before returnig from GMRES (with restart if the true residual does not indicate convergence). This option allows users to skip the evaluation and the check of the actual residual for badly conditioned problems where restart is not expected to be beneficial.

2.2.3.108 HYPRE_GMRESSetStopCrit()

```
HYPRE_Int HYPRE_GMRESSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.2.3.109 HYPRE_GMRESSetTol()

```
HYPRE_Int HYPRE_GMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the relative convergence tolerance.

2.2.3.110 HYPRE_GMRESSetup()

```
HYPRE_Int HYPRE_GMRESSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.111 HYPRE_GMRESSolve()

```
HYPRE_Int HYPRE_GMRESSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.2.3.112 HYPRE_LGMRESGetAugDim()

```
HYPRE_Int HYPRE_LGMRESGetAugDim (
            HYPRE_Solver solver,
            HYPRE_Int * k_dim )
```

2.2.3.113 HYPRE_LGMRESGetConverged()

```
HYPRE_Int HYPRE_LGMRESGetConverged (
            HYPRE_Solver solver,
            HYPRE_Int * converged )
```

2.2.3.114 HYPRE_LGMRESGetConvergenceFactorTol()

```
HYPRE_Int HYPRE_LGMRESGetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real * cf_tol )
```

2.2.3.115 HYPRE_LGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_LGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.116 HYPRE_LGMRESGetKDim()

```
HYPRE_Int HYPRE_LGMRESGetKDim (
            HYPRE_Solver solver,
            HYPRE_Int * k_dim )
```

2.2.3.117 HYPRE_LGMRESGetLogging()

```
HYPRE_Int HYPRE_LGMRESGetLogging (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.118 HYPRE_LGMRESGetMaxIter()

```
HYPRE_Int HYPRE_LGMRESGetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int * max_iter )
```

2.2.3.119 HYPRE_LGMRESGetMinIter()

```
HYPRE_Int HYPRE_LGMRESGetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int * min_iter )
```

2.2.3.120 HYPRE_LGMRESGetNumIterations()

```
HYPRE_Int HYPRE_LGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.121 HYPRE_LGMRESGetPrecond()

```
HYPRE_Int HYPRE_LGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.122 HYPRE_LGMRESGetPrintLevel()

```
HYPRE_Int HYPRE_LGMRESGetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.123 HYPRE_LGMRESGetResidual()

```
HYPRE_Int HYPRE_LGMRESGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.124 HYPRE_LGMRESGetStopCrit()

```
HYPRE_Int HYPRE_LGMRESGetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int * stop_crit )
```

2.2.3.125 HYPRE_LGMRESGetTol()

```
HYPRE_Int HYPRE_LGMRESGetTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.126 HYPRE_LGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_LGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The convergence test is $|r| \leq$ max(relative$_$tolerance$\ast |b|$, absolute$_$tolerance).)

2.2.3.127 HYPRE_LGMRESSetAugDim()

```
HYPRE_Int HYPRE_LGMRESSetAugDim (
            HYPRE_Solver solver,
            HYPRE_Int aug_dim )
```

(Optional) Set the number of augmentation vectors (default: 2).

2.2.3.128 HYPRE_LGMRESSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_LGMRESSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.129 HYPRE_LGMRESSetKDim()

```
HYPRE_Int HYPRE_LGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

(Optional) Set the maximum size of the approximation space (includes the augmentation vectors).

2.2.3.130 HYPRE_LGMRESSetLogging()

```
HYPRE_Int HYPRE_LGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.131 HYPRE_LGMRESSetMaxIter()

```
HYPRE_Int HYPRE_LGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.132 HYPRE_LGMRESSetMinIter()

```
HYPRE_Int HYPRE_LGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.2.3.133 HYPRE_LGMRESSetPrecond()

```
HYPRE_Int HYPRE_LGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.134 HYPRE_LGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_LGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.135 HYPRE_LGMRESSetTol()

```
HYPRE_Int HYPRE_LGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.2.3.136 HYPRE_LGMRESSetup()

```
HYPRE_Int HYPRE_LGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.137 HYPRE_LGMRESSolve()

```
HYPRE_Int HYPRE_LGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system. Details on LGMRES may be found in A. H. Baker, E.R. Jessup, and T.A. Manteuffel, "A technique for accelerating the convergence of restarted GMRES." SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 962-984. LGMRES(m,k) in the paper corresponds to LGMRES(Kdim+AugDim, AugDim).

2.2.3.138 HYPRE_PCGGetAbsoluteTolFactor()

```
HYPRE_Int HYPRE_PCGGetAbsoluteTolFactor (
            HYPRE_Solver solver,
            HYPRE_Real * abstolf )
```

2.2.3.139 HYPRE_PCGGetConverged()

```
HYPRE_Int HYPRE_PCGGetConverged (
            HYPRE_Solver solver,
            HYPRE_Int * converged )
```

2.2.3.140 HYPRE_PCGGetConvergenceFactorTol()

```
HYPRE_Int HYPRE_PCGGetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real * cf_tol )
```

2.2.3.141 HYPRE_PCGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_PCGGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.2.3.142 HYPRE_PCGGetLogging()

```
HYPRE_Int HYPRE_PCGGetLogging (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.143 HYPRE_PCGGetMaxIter()

```
HYPRE_Int HYPRE_PCGGetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int * max_iter )
```

2.2.3.144 HYPRE_PCGGetNumIterations()

```
HYPRE_Int HYPRE_PCGGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.2.3.145 HYPRE_PCGGetPrecond()

```
HYPRE_Int HYPRE_PCGGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.2.3.146 HYPRE_PCGGetPrintLevel()

```
HYPRE_Int HYPRE_PCGGetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int * level )
```

2.2.3.147 HYPRE_PCGGetRelChange()

```
HYPRE_Int HYPRE_PCGGetRelChange (
            HYPRE_Solver solver,
            HYPRE_Int * rel_change )
```

2.2.3.148 HYPRE_PCGGetResidual()

```
HYPRE_Int HYPRE_PCGGetResidual (
            HYPRE_Solver solver,
            void * residual )
```

Return the residual.

2.2.3.149 HYPRE_PCGGetResidualTol()

```
HYPRE_Int HYPRE_PCGGetResidualTol (
            HYPRE_Solver solver,
            HYPRE_Real * rtol )
```

2.2.3.150 HYPRE_PCGGetStopCrit()

```
HYPRE_Int HYPRE_PCGGetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int * stop_crit )
```

2.2.3.151 HYPRE_PCGGetTol()

```
HYPRE_Int HYPRE_PCGGetTol (
            HYPRE_Solver solver,
            HYPRE_Real * tol )
```

2.2.3.152 HYPRE_PCGGetTwoNorm()

```
HYPRE_Int HYPRE_PCGGetTwoNorm (
            HYPRE_Solver solver,
            HYPRE_Int * two_norm )
```

2.2.3.153 HYPRE_PCGSetAbsoluteTol()

```
HYPRE_Int HYPRE_PCGSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

(Optional) Set the absolute convergence tolerance (default is 0). If one desires the convergence test to check the absolute convergence tolerance {\it only}, then set the relative convergence tolerance to 0.0. (The default convergence test is $ <C*r,r> \leq$ max(relative$_$tolerance$^{2} \ast <C*b, b>$, absolute$_$tolerance^2).)

2.2.3.154 HYPRE_PCGSetAbsoluteTolFactor()

```
HYPRE_Int HYPRE_PCGSetAbsoluteTolFactor (
            HYPRE_Solver solver,
            HYPRE_Real abstolf )
```

2.2.3.155 HYPRE_PCGSetConvergenceFactorTol()

```
HYPRE_Int HYPRE_PCGSetConvergenceFactorTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

2.2.3.156 HYPRE_PCGSetLogging()

```
HYPRE_Int HYPRE_PCGSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.2.3.157 HYPRE_PCGSetMaxIter()

```
HYPRE_Int HYPRE_PCGSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.2.3.158 HYPRE_PCGSetPrecond()

```
HYPRE_Int HYPRE_PCGSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use.

2.2.3.159 HYPRE_PCGSetPrintLevel()

```
HYPRE_Int HYPRE_PCGSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.2.3.160 HYPRE_PCGSetRecomputeResidual()

```
HYPRE_Int HYPRE_PCGSetRecomputeResidual (
             HYPRE_Solver solver,
             HYPRE_Int recompute_residual )
```

(Optional) Recompute the residual at the end to double-check convergence.

2.2.3.161 HYPRE_PCGSetRecomputeResidualP()

```
HYPRE_Int HYPRE_PCGSetRecomputeResidualP (
             HYPRE_Solver solver,
             HYPRE_Int recompute_residual_p )
```

(Optional) Periodically recompute the residual while iterating.

2.2.3.162 HYPRE_PCGSetRelChange()

```
HYPRE_Int HYPRE_PCGSetRelChange (
             HYPRE_Solver solver,
             HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.2.3.163 HYPRE_PCGSetResidualTol()

```
HYPRE_Int HYPRE_PCGSetResidualTol (
             HYPRE_Solver solver,
             HYPRE_Real rtol )
```

(Optional) Set a residual-based convergence tolerance which checks if $|r_{old}-r_{new}| < rtol |b|$. This is useful when trying to converge to very low relative and/or absolute tolerances, in order to bail-out before roundoff errors affect the approximation.

2.2.3.164 HYPRE_PCGSetStopCrit()

```
HYPRE_Int HYPRE_PCGSetStopCrit (
             HYPRE_Solver solver,
             HYPRE_Int stop_crit )
```

2.2.3.165 HYPRE_PCGSetTol()

```
HYPRE_Int HYPRE_PCGSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the relative convergence tolerance.

2.2.3.166 HYPRE_PCGSetTwoNorm()

```
HYPRE_Int HYPRE_PCGSetTwoNorm (
            HYPRE_Solver solver,
            HYPRE_Int two_norm )
```

(Optional) Use the two-norm in stopping criteria.

2.2.3.167 HYPRE_PCGSetup()

```
HYPRE_Int HYPRE_PCGSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.2.3.168 HYPRE_PCGSolve()

```
HYPRE_Int HYPRE_PCGSolve (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Solve the system.

2.3 HYPRE_lobpcg.h File Reference

Functions

LOBPCG Eigensolver

- HYPRE_Int HYPRE_LOBPCGCreate (mv_InterfaceInterpreter *interpreter, HYPRE_MatvecFunctions *mvfunctions, HYPRE_Solver *solver)
- HYPRE_Int HYPRE_LOBPCGDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_LOBPCGSetPrecond (HYPRE_Solver solver, HYPRE_PtrToSolverFcn precond, HYPRE_PtrToSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_LOBPCGGetPrecond (HYPRE_Solver solver, HYPRE_Solver *precond_data_ptr)
- HYPRE_Int HYPRE_LOBPCGSetup (HYPRE_Solver solver, HYPRE_Matrix A, HYPRE_Vector b, HYPRE_Vector x)
- HYPRE_Int HYPRE_LOBPCGSetupB (HYPRE_Solver solver, HYPRE_Matrix B, HYPRE_Vector x)
- HYPRE_Int HYPRE_LOBPCGSetupT (HYPRE_Solver solver, HYPRE_Matrix T, HYPRE_Vector x)
- HYPRE_Int HYPRE_LOBPCGSolve (HYPRE_Solver solver, mv_MultiVectorPtr y, mv_MultiVectorPtr x, HYPRE_Real *lambda)
- HYPRE_Int HYPRE_LOBPCGSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_LOBPCGSetRTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_LOBPCGSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_LOBPCGSetPrecondUsageMode (HYPRE_Solver solver, HYPRE_Int mode)
- HYPRE_Int HYPRE_LOBPCGSetPrintLevel (HYPRE_Solver solver, HYPRE_Int level)
- utilities_FortranMatrix * HYPRE_LOBPCGResidualNorms (HYPRE_Solver solver)
- utilities_FortranMatrix * HYPRE_LOBPCGResidualNormsHistory (HYPRE_Solver solver)
- utilities_FortranMatrix * HYPRE_LOBPCGEigenvaluesHistory (HYPRE_Solver solver)
- HYPRE_Int HYPRE_LOBPCGIterations (HYPRE_Solver solver)
- void hypre_LOBPCGMultiOperatorB (void *data, void *x, void *y)
- void lobpcg_MultiVectorByMultiVector (mv_MultiVectorPtr x, mv_MultiVectorPtr y, utilities_FortranMatrix *xy)

2.3.1 Function Documentation

2.3.1.1 HYPRE_LOBPCGCreate()

```
HYPRE_Int HYPRE_LOBPCGCreate (
            mv_InterfaceInterpreter * interpreter,
            HYPRE_MatvecFunctions * mvfunctions,
            HYPRE_Solver * solver )
```

LOBPCG constructor.

2.3.1.2 HYPRE_LOBPCGDestroy()

```
HYPRE_Int HYPRE_LOBPCGDestroy (
            HYPRE_Solver solver )
```

LOBPCG destructor.

2.3.1.3 HYPRE_LOBPCGEigenvaluesHistory()

```
utilities_FortranMatrix* HYPRE_LOBPCGEigenvaluesHistory (
            HYPRE_Solver solver )
```

2.3.1.4 HYPRE_LOBPCGGetPrecond()

```
HYPRE_Int HYPRE_LOBPCGGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data_ptr )
```

2.3.1.5 HYPRE_LOBPCGIterations()

```
HYPRE_Int HYPRE_LOBPCGIterations (
            HYPRE_Solver solver )
```

2.3.1.6 hypre_LOBPCGMultiOperatorB()

```
void hypre_LOBPCGMultiOperatorB (
            void * data,
            void * x,
            void * y )
```

2.3.1.7 HYPRE_LOBPCGResidualNorms()

```
utilities_FortranMatrix* HYPRE_LOBPCGResidualNorms (
            HYPRE_Solver solver )
```

2.3.1.8 HYPRE_LOBPCGResidualNormsHistory()

```
utilities_FortranMatrix* HYPRE_LOBPCGResidualNormsHistory (
            HYPRE_Solver solver )
```

2.3.1.9 HYPRE_LOBPCGSetMaxIter()

```
HYPRE_Int HYPRE_LOBPCGSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.3.1.10 HYPRE_LOBPCGSetPrecond()

```
HYPRE_Int HYPRE_LOBPCGSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToSolverFcn precond,
            HYPRE_PtrToSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

(Optional) Set the preconditioner to use. If not called, preconditioning is not used.

2.3.1.11 HYPRE_LOBPCGSetPrecondUsageMode()

```
HYPRE_Int HYPRE_LOBPCGSetPrecondUsageMode (
            HYPRE_Solver solver,
            HYPRE_Int mode )
```

Define which initial guess for inner PCG iterations to use: {\tt mode} = 0: use zero initial guess, otherwise use RHS.

2.3.1.12 HYPRE_LOBPCGSetPrintLevel()

```
HYPRE_Int HYPRE_LOBPCGSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Set the amount of printing to do to the screen.

2.3.1.13 HYPRE_LOBPCGSetRTol()

```
HYPRE_Int HYPRE_LOBPCGSetRTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the relative convergence tolerance.

2.3.1.14 HYPRE_LOBPCGSetTol()

```
HYPRE_Int HYPRE_LOBPCGSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the absolute convergence tolerance.

2.3.1.15 HYPRE_LOBPCGSetup()

```
HYPRE_Int HYPRE_LOBPCGSetup (
            HYPRE_Solver solver,
            HYPRE_Matrix A,
            HYPRE_Vector b,
            HYPRE_Vector x )
```

Set up {\tt A} and the preconditioner (if there is one).

2.3.1.16 HYPRE_LOBPCGSetupB()

```
HYPRE_Int HYPRE_LOBPCGSetupB (
            HYPRE_Solver solver,
            HYPRE_Matrix B,
            HYPRE_Vector x )
```

(Optional) Set up {\tt B}. If not called, B = I.

2.3.1.17 HYPRE_LOBPCGSetupT()

```
HYPRE_Int HYPRE_LOBPCGSetupT (
            HYPRE_Solver solver,
            HYPRE_Matrix T,
            HYPRE_Vector x )
```

(Optional) Set the preconditioning to be applied to Tx = b, not Ax = b.

2.3.1.18 HYPRE_LOBPCGSolve()

```
HYPRE_Int HYPRE_LOBPCGSolve (
            HYPRE_Solver solver,
            mv_MultiVectorPtr y,
            mv_MultiVectorPtr x,
            HYPRE_Real * lambda )
```

Solve A x = lambda B x, y'x = 0.

2.3.1.19 lobpcg_MultiVectorByMultiVector()

```
void lobpcg_MultiVectorByMultiVector (
            mv_MultiVectorPtr x,
            mv_MultiVectorPtr y,
            utilities_FortranMatrix * xy )
```

2.4 HYPRE_parcsr_ls.h File Reference

Functions

- HYPRE_Int HYPRE_AMSSetInterpolations (HYPRE_Solver solver, HYPRE_ParCSRMatrix Pi, HYPRE_Par↩ CSRMatrix Pix, HYPRE_ParCSRMatrix Piy, HYPRE_ParCSRMatrix Piz)
- HYPRE_Int HYPRE_AMSSetAlphaPoissonMatrix (HYPRE_Solver solver, HYPRE_ParCSRMatrix A_alpha)
- HYPRE_Int HYPRE_AMSSetBetaPoissonMatrix (HYPRE_Solver solver, HYPRE_ParCSRMatrix A_beta)
- HYPRE_Int HYPRE_AMSSetInteriorNodes (HYPRE_Solver solver, HYPRE_ParVector interior_nodes)
- HYPRE_Int HYPRE_AMSSetProjectionFrequency (HYPRE_Solver solver, HYPRE_Int projection_frequency)
- HYPRE_Int HYPRE_AMSSetMaxIter (HYPRE_Solver solver, HYPRE_Int maxit)
- HYPRE_Int HYPRE_AMSSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_AMSSetCycleType (HYPRE_Solver solver, HYPRE_Int cycle_type)
- HYPRE_Int HYPRE_AMSSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_AMSSetSmoothingOptions (HYPRE_Solver solver, HYPRE_Int relax_type, HYPRE_Int relax_times, HYPRE_Real relax_weight, HYPRE_Real omega)
- HYPRE_Int HYPRE_AMSSetAlphaAMGOptions (HYPRE_Solver solver, HYPRE_Int alpha_coarsen_type, HYPRE_Int alpha_agg_levels, HYPRE_Int alpha_relax_type, HYPRE_Real alpha_strength_threshold, HYPRE↩ _Int alpha_interp_type, HYPRE_Int alpha_Pmax)
- HYPRE_Int HYPRE_AMSSetAlphaAMGCoarseRelaxType (HYPRE_Solver solver, HYPRE_Int alpha_coarse_↩ relax_type)
- HYPRE_Int HYPRE_AMSSetBetaAMGOptions (HYPRE_Solver solver, HYPRE_Int beta_coarsen_type, HYPRE_Int beta_agg_levels, HYPRE_Int beta_relax_type, HYPRE_Real beta_strength_threshold, HYPRE_Int beta_interp_type, HYPRE_Int beta_Pmax)
- HYPRE_Int HYPRE_AMSSetBetaAMGCoarseRelaxType (HYPRE_Solver solver, HYPRE_Int beta_coarse_↩ relax_type)
- HYPRE_Int HYPRE_AMSGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_AMSGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗rel_resid_↩ norm)
- HYPRE_Int HYPRE_AMSProjectOutGradients (HYPRE_Solver solver, HYPRE_ParVector x)
- HYPRE_Int HYPRE_AMSConstructDiscreteGradient (HYPRE_ParCSRMatrix A, HYPRE_ParVector x_coord, HYPRE_BigInt ∗edge_vertex, HYPRE_Int edge_orientation, HYPRE_ParCSRMatrix ∗G)
- HYPRE_Int HYPRE_ADSSetInterpolations (HYPRE_Solver solver, HYPRE_ParCSRMatrix RT_Pi, HYPRE_↩ ParCSRMatrix RT_Pix, HYPRE_ParCSRMatrix RT_Piy, HYPRE_ParCSRMatrix RT_Piz, HYPRE_ParCSRMatrix ND_Pi, HYPRE_ParCSRMatrix ND_Pix, HYPRE_ParCSRMatrix ND_Piy, HYPRE_ParCSRMatrix ND_Piz)
- HYPRE_Int HYPRE_ADSSetMaxIter (HYPRE_Solver solver, HYPRE_Int maxit)
- HYPRE_Int HYPRE_ADSSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ADSSetCycleType (HYPRE_Solver solver, HYPRE_Int cycle_type)
- HYPRE_Int HYPRE_ADSSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ADSSetSmoothingOptions (HYPRE_Solver solver, HYPRE_Int relax_type, HYPRE_Int relax_times, HYPRE_Real relax_weight, HYPRE_Real omega)

- HYPRE_Int HYPRE_ADSSetChebySmoothingOptions (HYPRE_Solver solver, HYPRE_Int cheby_order, HYPRE_Int cheby_fraction)
- HYPRE_Int HYPRE_ADSSetAMSOptions (HYPRE_Solver solver, HYPRE_Int cycle_type, HYPRE_Int coarsen←_type, HYPRE_Int agg_levels, HYPRE_Int relax_type, HYPRE_Real strength_threshold, HYPRE_Int interp_type, HYPRE_Int Pmax)
- HYPRE_Int HYPRE_ADSSetAMGOptions (HYPRE_Solver solver, HYPRE_Int coarsen_type, HYPRE_Int agg←_levels, HYPRE_Int relax_type, HYPRE_Real strength_threshold, HYPRE_Int interp_type, HYPRE_Int Pmax)
- HYPRE_Int HYPRE_ADSGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ADSGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗rel_resid_norm)

ParCSR BoomerAMG Solver and Preconditioner

Parallel unstructured algebraic multigrid solver and preconditioner

- HYPRE_Int HYPRE_BoomerAMGCreate (HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_BoomerAMGDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_BoomerAMGSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_Par←Vector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_BoomerAMGSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_BoomerAMGSolveT (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_Par←Vector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_BoomerAMGSetOldDefault (HYPRE_Solver solver)
- HYPRE_Int HYPRE_BoomerAMGGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)
- HYPRE_Int HYPRE_BoomerAMGGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_BoomerAMGGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗rel_resid_norm)
- HYPRE_Int HYPRE_BoomerAMGSetNumFunctions (HYPRE_Solver solver, HYPRE_Int num_functions)
- HYPRE_Int HYPRE_BoomerAMGSetDofFunc (HYPRE_Solver solver, HYPRE_Int ∗dof_func)
- HYPRE_Int HYPRE_BoomerAMGSetConvergeType (HYPRE_Solver solver, HYPRE_Int type)
- HYPRE_Int HYPRE_BoomerAMGSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_BoomerAMGSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_BoomerAMGSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_BoomerAMGSetMaxCoarseSize (HYPRE_Solver solver, HYPRE_Int max_coarse_size)
- HYPRE_Int HYPRE_BoomerAMGSetMinCoarseSize (HYPRE_Solver solver, HYPRE_Int min_coarse_size)
- HYPRE_Int HYPRE_BoomerAMGSetMaxLevels (HYPRE_Solver solver, HYPRE_Int max_levels)
- HYPRE_Int HYPRE_BoomerAMGSetStrongThreshold (HYPRE_Solver solver, HYPRE_Real strong_←threshold)
- HYPRE_Int HYPRE_BoomerAMGSetStrongThresholdR (HYPRE_Solver solver, HYPRE_Real strong_←threshold)
- HYPRE_Int HYPRE_BoomerAMGSetFilterThresholdR (HYPRE_Solver solver, HYPRE_Real filter_threshold)
- HYPRE_Int HYPRE_BoomerAMGSetSCommPkgSwitch (HYPRE_Solver solver, HYPRE_Real S_commpkg←_switch)
- HYPRE_Int HYPRE_BoomerAMGSetMaxRowSum (HYPRE_Solver solver, HYPRE_Real max_row_sum)
- HYPRE_Int HYPRE_BoomerAMGSetCoarsenType (HYPRE_Solver solver, HYPRE_Int coarsen_type)
- HYPRE_Int HYPRE_BoomerAMGSetNonGalerkinTol (HYPRE_Solver solver, HYPRE_Real nongalerkin_tol)
- HYPRE_Int HYPRE_BoomerAMGSetLevelNonGalerkinTol (HYPRE_Solver solver, HYPRE_Real nongalerkin←_tol, HYPRE_Int level)
- HYPRE_Int HYPRE_BoomerAMGSetNonGalerkTol (HYPRE_Solver solver, HYPRE_Int nongalerk_num_tol, HYPRE_Real ∗nongalerk_tol)
- HYPRE_Int HYPRE_BoomerAMGSetMeasureType (HYPRE_Solver solver, HYPRE_Int measure_type)
- HYPRE_Int HYPRE_BoomerAMGSetAggNumLevels (HYPRE_Solver solver, HYPRE_Int agg_num_levels)
- HYPRE_Int HYPRE_BoomerAMGSetNumPaths (HYPRE_Solver solver, HYPRE_Int num_paths)
- HYPRE_Int HYPRE_BoomerAMGSetCGCIts (HYPRE_Solver solver, HYPRE_Int its)
- HYPRE_Int HYPRE_BoomerAMGSetNodal (HYPRE_Solver solver, HYPRE_Int nodal)
- HYPRE_Int HYPRE_BoomerAMGSetNodalDiag (HYPRE_Solver solver, HYPRE_Int nodal_diag)
- HYPRE_Int HYPRE_BoomerAMGSetInterpType (HYPRE_Solver solver, HYPRE_Int interp_type)

- HYPRE_Int HYPRE_BoomerAMGSetTruncFactor (HYPRE_Solver solver, HYPRE_Real trunc_factor)
- HYPRE_Int HYPRE_BoomerAMGSetPMaxElmts (HYPRE_Solver solver, HYPRE_Int P_max_elmts)
- HYPRE_Int HYPRE_BoomerAMGSetSepWeight (HYPRE_Solver solver, HYPRE_Int sep_weight)
- HYPRE_Int HYPRE_BoomerAMGSetAggInterpType (HYPRE_Solver solver, HYPRE_Int agg_interp_type)
- HYPRE_Int HYPRE_BoomerAMGSetAggTruncFactor (HYPRE_Solver solver, HYPRE_Real agg_trunc_factor)
- HYPRE_Int HYPRE_BoomerAMGSetAggP12TruncFactor (HYPRE_Solver solver, HYPRE_Real agg_P12_↩ trunc_factor)
- HYPRE_Int HYPRE_BoomerAMGSetAggPMaxElmts (HYPRE_Solver solver, HYPRE_Int agg_P_max_elmts)
- HYPRE_Int HYPRE_BoomerAMGSetAggP12MaxElmts (HYPRE_Solver solver, HYPRE_Int agg_P12_max↩ _elmts)
- HYPRE_Int HYPRE_BoomerAMGSetInterpVectors (HYPRE_Solver solver, HYPRE_Int num_vectors, HYPRE_ParVector ∗interp_vectors)
- HYPRE_Int HYPRE_BoomerAMGSetInterpVecVariant (HYPRE_Solver solver, HYPRE_Int var)
- HYPRE_Int HYPRE_BoomerAMGSetInterpVecQMax (HYPRE_Solver solver, HYPRE_Int q_max)
- HYPRE_Int HYPRE_BoomerAMGSetInterpVecAbsQTrunc (HYPRE_Solver solver, HYPRE_Real q_trunc)
- HYPRE_Int HYPRE_BoomerAMGSetGSMG (HYPRE_Solver solver, HYPRE_Int gsmg)
- HYPRE_Int HYPRE_BoomerAMGSetNumSamples (HYPRE_Solver solver, HYPRE_Int num_samples)
- HYPRE_Int HYPRE_BoomerAMGSetCycleType (HYPRE_Solver solver, HYPRE_Int cycle_type)
- HYPRE_Int HYPRE_BoomerAMGSetFCycle (HYPRE_Solver solver, HYPRE_Int fcycle)
- HYPRE_Int HYPRE_BoomerAMGSetAdditive (HYPRE_Solver solver, HYPRE_Int addlvl)
- HYPRE_Int HYPRE_BoomerAMGSetMultAdditive (HYPRE_Solver solver, HYPRE_Int addlvl)
- HYPRE_Int HYPRE_BoomerAMGSetSimple (HYPRE_Solver solver, HYPRE_Int addlvl)
- HYPRE_Int HYPRE_BoomerAMGSetAddLastLvl (HYPRE_Solver solver, HYPRE_Int add_last_lvl)
- HYPRE_Int HYPRE_BoomerAMGSetMultAddTruncFactor (HYPRE_Solver solver, HYPRE_Real add_trunc↩ _factor)
- HYPRE_Int HYPRE_BoomerAMGSetMultAddPMaxElmts (HYPRE_Solver solver, HYPRE_Int add_P_max_↩ elmts)
- HYPRE_Int HYPRE_BoomerAMGSetAddRelaxType (HYPRE_Solver solver, HYPRE_Int add_rlx_type)
- HYPRE_Int HYPRE_BoomerAMGSetAddRelaxWt (HYPRE_Solver solver, HYPRE_Real add_rlx_wt)
- HYPRE_Int HYPRE_BoomerAMGSetSeqThreshold (HYPRE_Solver solver, HYPRE_Int seq_threshold)
- HYPRE_Int HYPRE_BoomerAMGSetRedundant (HYPRE_Solver solver, HYPRE_Int redundant)
- HYPRE_Int HYPRE_BoomerAMGSetNumGridSweeps (HYPRE_Solver solver, HYPRE_Int ∗num_grid_↩ sweeps)
- HYPRE_Int HYPRE_BoomerAMGSetNumSweeps (HYPRE_Solver solver, HYPRE_Int num_sweeps)
- HYPRE_Int HYPRE_BoomerAMGSetCycleNumSweeps (HYPRE_Solver solver, HYPRE_Int num_sweeps, HYPRE_Int k)
- HYPRE_Int HYPRE_BoomerAMGSetGridRelaxType (HYPRE_Solver solver, HYPRE_Int ∗grid_relax_type)
- HYPRE_Int HYPRE_BoomerAMGSetRelaxType (HYPRE_Solver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_BoomerAMGSetCycleRelaxType (HYPRE_Solver solver, HYPRE_Int relax_type, HYPRE_Int k)
- HYPRE_Int HYPRE_BoomerAMGSetRelaxOrder (HYPRE_Solver solver, HYPRE_Int relax_order)
- HYPRE_Int HYPRE_BoomerAMGSetGridRelaxPoints (HYPRE_Solver solver, HYPRE_Int ∗∗grid_relax_↩ points)
- HYPRE_Int HYPRE_BoomerAMGSetRelaxWeight (HYPRE_Solver solver, HYPRE_Real ∗relax_weight)
- HYPRE_Int HYPRE_BoomerAMGSetRelaxWt (HYPRE_Solver solver, HYPRE_Real relax_weight)
- HYPRE_Int HYPRE_BoomerAMGSetLevelRelaxWt (HYPRE_Solver solver, HYPRE_Real relax_weight, HYPRE_Int level)
- HYPRE_Int HYPRE_BoomerAMGSetOmega (HYPRE_Solver solver, HYPRE_Real ∗omega)
- HYPRE_Int HYPRE_BoomerAMGSetOuterWt (HYPRE_Solver solver, HYPRE_Real omega)
- HYPRE_Int HYPRE_BoomerAMGSetLevelOuterWt (HYPRE_Solver solver, HYPRE_Real omega, HYPRE_Int level)
- HYPRE_Int HYPRE_BoomerAMGSetChebyOrder (HYPRE_Solver solver, HYPRE_Int order)
- HYPRE_Int HYPRE_BoomerAMGSetChebyFraction (HYPRE_Solver solver, HYPRE_Real ratio)
- HYPRE_Int HYPRE_BoomerAMGSetChebyScale (HYPRE_Solver solver, HYPRE_Int scale)
- HYPRE_Int HYPRE_BoomerAMGSetChebyVariant (HYPRE_Solver solver, HYPRE_Int variant)
- HYPRE_Int HYPRE_BoomerAMGSetChebyEigEst (HYPRE_Solver solver, HYPRE_Int eig_est)
- HYPRE_Int HYPRE_BoomerAMGSetSmoothType (HYPRE_Solver solver, HYPRE_Int smooth_type)
- HYPRE_Int HYPRE_BoomerAMGSetSmoothNumLevels (HYPRE_Solver solver, HYPRE_Int smooth_num↩ _levels)

- HYPRE_Int HYPRE_BoomerAMGSetSmoothNumSweeps (HYPRE_Solver solver, HYPRE_Int smooth_num_sweeps)
- HYPRE_Int HYPRE_BoomerAMGSetVariant (HYPRE_Solver solver, HYPRE_Int variant)
- HYPRE_Int HYPRE_BoomerAMGSetOverlap (HYPRE_Solver solver, HYPRE_Int overlap)
- HYPRE_Int HYPRE_BoomerAMGSetDomainType (HYPRE_Solver solver, HYPRE_Int domain_type)
- HYPRE_Int HYPRE_BoomerAMGSetSchwarzRlxWeight (HYPRE_Solver solver, HYPRE_Real schwarz_rlx_weight)
- HYPRE_Int HYPRE_BoomerAMGSetSchwarzUseNonSymm (HYPRE_Solver solver, HYPRE_Int use_nonsymm)
- HYPRE_Int HYPRE_BoomerAMGSetSym (HYPRE_Solver solver, HYPRE_Int sym)
- HYPRE_Int HYPRE_BoomerAMGSetLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_BoomerAMGSetThreshold (HYPRE_Solver solver, HYPRE_Real threshold)
- HYPRE_Int HYPRE_BoomerAMGSetFilter (HYPRE_Solver solver, HYPRE_Real filter)
- HYPRE_Int HYPRE_BoomerAMGSetDropTol (HYPRE_Solver solver, HYPRE_Real drop_tol)
- HYPRE_Int HYPRE_BoomerAMGSetMaxNzPerRow (HYPRE_Solver solver, HYPRE_Int max_nz_per_row)
- HYPRE_Int HYPRE_BoomerAMGSetEuclidFile (HYPRE_Solver solver, char ∗euclidfile)
- HYPRE_Int HYPRE_BoomerAMGSetEuLevel (HYPRE_Solver solver, HYPRE_Int eu_level)
- HYPRE_Int HYPRE_BoomerAMGSetEuSparseA (HYPRE_Solver solver, HYPRE_Real eu_sparse_A)
- HYPRE_Int HYPRE_BoomerAMGSetEuBJ (HYPRE_Solver solver, HYPRE_Int eu_bj)
- HYPRE_Int HYPRE_BoomerAMGSetRestriction (HYPRE_Solver solver, HYPRE_Int restr_par)
- HYPRE_Int HYPRE_BoomerAMGSetIsTriangular (HYPRE_Solver solver, HYPRE_Int is_triangular)
- HYPRE_Int HYPRE_BoomerAMGSetGMRESSwitchR (HYPRE_Solver solver, HYPRE_Int gmres_switch)
- HYPRE_Int HYPRE_BoomerAMGSetADropTol (HYPRE_Solver solver, HYPRE_Real A_drop_tol)
- HYPRE_Int HYPRE_BoomerAMGSetADropType (HYPRE_Solver solver, HYPRE_Int A_drop_type)
- HYPRE_Int HYPRE_BoomerAMGSetPrintFileName (HYPRE_Solver solver, const char ∗print_file_name)
- HYPRE_Int HYPRE_BoomerAMGSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_BoomerAMGSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_BoomerAMGSetDebugFlag (HYPRE_Solver solver, HYPRE_Int debug_flag)
- HYPRE_Int HYPRE_BoomerAMGInitGridRelaxation (HYPRE_Int ∗∗num_grid_sweeps_ptr, HYPRE_Int ∗∗grid_relax_type_ptr, HYPRE_Int ∗∗∗grid_relax_points_ptr, HYPRE_Int coarsen_type, HYPRE_Real ∗∗relax_weights_ptr, HYPRE_Int max_levels)
- HYPRE_Int HYPRE_BoomerAMGSetRAP2 (HYPRE_Solver solver, HYPRE_Int rap2)
- HYPRE_Int HYPRE_BoomerAMGSetModuleRAP2 (HYPRE_Solver solver, HYPRE_Int mod_rap2)
- HYPRE_Int HYPRE_BoomerAMGSetKeepTranspose (HYPRE_Solver solver, HYPRE_Int keepTranspose)
- HYPRE_Int HYPRE_BoomerAMGSetPlotGrids (HYPRE_Solver solver, HYPRE_Int plotgrids)
- HYPRE_Int HYPRE_BoomerAMGSetPlotFileName (HYPRE_Solver solver, const char ∗plotfilename)
- HYPRE_Int HYPRE_BoomerAMGSetCoordDim (HYPRE_Solver solver, HYPRE_Int coorddim)
- HYPRE_Int HYPRE_BoomerAMGSetCoordinates (HYPRE_Solver solver, float ∗coordinates)
- HYPRE_Int HYPRE_BoomerAMGSetCpointsToKeep (HYPRE_Solver solver, HYPRE_Int cpt_coarse_level, HYPRE_Int num_cpt_coarse, HYPRE_Int ∗cpt_coarse_index)
- HYPRE_Int HYPRE_BoomerAMGSetSabs (HYPRE_Solver solver, HYPRE_Int Sabs)

ParCSR ParaSails Preconditioner

Parallel sparse approximate inverse preconditioner for the ParCSR matrix format.

- HYPRE_Int HYPRE_ParaSailsCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParaSailsDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParaSailsSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParaSailsSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParaSailsSetParams (HYPRE_Solver solver, HYPRE_Real thresh, HYPRE_Int nlevels)
- HYPRE_Int HYPRE_ParaSailsSetFilter (HYPRE_Solver solver, HYPRE_Real filter)
- HYPRE_Int HYPRE_ParaSailsSetSym (HYPRE_Solver solver, HYPRE_Int sym)
- HYPRE_Int HYPRE_ParaSailsSetLoadbal (HYPRE_Solver solver, HYPRE_Real loadbal)
- HYPRE_Int HYPRE_ParaSailsSetReuse (HYPRE_Solver solver, HYPRE_Int reuse)
- HYPRE_Int HYPRE_ParaSailsSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParaSailsBuildIJMatrix (HYPRE_Solver solver, HYPRE_IJMatrix ∗pij_A)

- HYPRE_Int HYPRE_ParCSRParaSailsCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRParaSailsDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRParaSailsSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRParaSailsSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRParaSailsSetParams (HYPRE_Solver solver, HYPRE_Real thresh, HYPRE_Int nlevels)
- HYPRE_Int HYPRE_ParCSRParaSailsSetFilter (HYPRE_Solver solver, HYPRE_Real filter)
- HYPRE_Int HYPRE_ParCSRParaSailsSetSym (HYPRE_Solver solver, HYPRE_Int sym)
- HYPRE_Int HYPRE_ParCSRParaSailsSetLoadbal (HYPRE_Solver solver, HYPRE_Real loadbal)
- HYPRE_Int HYPRE_ParCSRParaSailsSetReuse (HYPRE_Solver solver, HYPRE_Int reuse)
- HYPRE_Int HYPRE_ParCSRParaSailsSetLogging (HYPRE_Solver solver, HYPRE_Int logging)

ParCSR Euclid Preconditioner

MPI Parallel ILU preconditioner

Options summary: \begin{center} \begin{tabular}{|l|c|l|} \hline Option & Default & Synopsis \ \hline -level & 1 & ILU(k) factorization level \ \hline -bj & 0 (false) & Use Block Jacobi ILU instead of PILU \ \hline -eu_stats & 0 (false) & Print internal timing and statistics \ \hline -eu_mem & 0 (false) & Print internal memory usage \ \hline \end{tabular} \end{center}

- HYPRE_Int HYPRE_EuclidCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_EuclidDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_EuclidSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_EuclidSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_EuclidSetParams (HYPRE_Solver solver, HYPRE_Int argc, char ∗argv[ ])
- HYPRE_Int HYPRE_EuclidSetParamsFromFile (HYPRE_Solver solver, char ∗filename)
- HYPRE_Int HYPRE_EuclidSetLevel (HYPRE_Solver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_EuclidSetBJ (HYPRE_Solver solver, HYPRE_Int bj)
- HYPRE_Int HYPRE_EuclidSetStats (HYPRE_Solver solver, HYPRE_Int eu_stats)
- HYPRE_Int HYPRE_EuclidSetMem (HYPRE_Solver solver, HYPRE_Int eu_mem)
- HYPRE_Int HYPRE_EuclidSetSparseA (HYPRE_Solver solver, HYPRE_Real sparse_A)
- HYPRE_Int HYPRE_EuclidSetRowScale (HYPRE_Solver solver, HYPRE_Int row_scale)
- HYPRE_Int HYPRE_EuclidSetILUT (HYPRE_Solver solver, HYPRE_Real drop_tol)

ParCSR Pilut Preconditioner

- HYPRE_Int HYPRE_ParCSRPilutCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRPilutDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRPilutSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRPilutSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRPilutSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRPilutSetDropTolerance (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRPilutSetFactorRowSize (HYPRE_Solver solver, HYPRE_Int size)

ParCSR AMS Solver and Preconditioner

Parallel auxiliary space Maxwell solver and preconditioner

- HYPRE_Int HYPRE_AMSCreate (HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_AMSDestroy (HYPRE_Solver solver)

- HYPRE_Int HYPRE_AMSSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_AMSSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_AMSSetDimension (HYPRE_Solver solver, HYPRE_Int dim)
- HYPRE_Int HYPRE_AMSSetDiscreteGradient (HYPRE_Solver solver, HYPRE_ParCSRMatrix G)
- HYPRE_Int HYPRE_AMSSetCoordinateVectors (HYPRE_Solver solver, HYPRE_ParVector x, HYPRE_ParVector y, HYPRE_ParVector z)
- HYPRE_Int HYPRE_AMSSetEdgeConstantVectors (HYPRE_Solver solver, HYPRE_ParVector Gx, HYPRE_ParVector Gy, HYPRE_ParVector Gz)

ParCSR ADS Solver and Preconditioner

Parallel auxiliary space divergence solver and preconditioner

- HYPRE_Int HYPRE_ADSCreate (HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ADSDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ADSSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ADSSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ADSSetDiscreteCurl (HYPRE_Solver solver, HYPRE_ParCSRMatrix C)
- HYPRE_Int HYPRE_ADSSetDiscreteGradient (HYPRE_Solver solver, HYPRE_ParCSRMatrix G)
- HYPRE_Int HYPRE_ADSSetCoordinateVectors (HYPRE_Solver solver, HYPRE_ParVector x, HYPRE_ParVector y, HYPRE_ParVector z)

ParCSR PCG Solver

These routines should be used in conjunction with the generic interface in \Ref{PCG Solver}.

- HYPRE_Int HYPRE_ParCSRPCGCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRPCGDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRPCGSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRPCGSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRPCGSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRPCGSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRPCGSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRPCGSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_ParCSRPCGSetTwoNorm (HYPRE_Solver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_ParCSRPCGSetRelChange (HYPRE_Solver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_ParCSRPCGSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRPCGGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRPCGSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRPCGSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRPCGGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRPCGGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRPCGGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)
- HYPRE_Int HYPRE_ParCSRDiagScaleSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector y, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRDiagScale (HYPRE_Solver solver, HYPRE_ParCSRMatrix HA, HYPRE_ParVector Hy, HYPRE_ParVector Hx)
- HYPRE_Int HYPRE_ParCSROnProcTriSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix HA, HYPRE_ParVector Hy, HYPRE_ParVector Hx)

- HYPRE_Int HYPRE_ParCSROnProcTriSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix HA, HYPRE_ParVector Hy, HYPRE_ParVector Hx)

ParCSR GMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{GMRES Solver}.

- HYPRE_Int HYPRE_ParCSRGMRESCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRGMRESDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRGMRESSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRGMRESSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_ParCSRGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_ParCSRGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRGMRESSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_ParCSRGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRGMRESGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)
- HYPRE_Int HYPRE_ParCSRCOGMRESCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRCOGMRESDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRCOGMRESSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetUnroll (HYPRE_Solver solver, HYPRE_Int unroll)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetCGS (HYPRE_Solver solver, HYPRE_Int cgs)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRCOGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRCOGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRCOGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRCOGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRCOGMRESGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)

ParCSR FlexGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{FlexGMRES Solver}.

- HYPRE_Int HYPRE_ParCSRFlexGMRESCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRFlexGMRESDestroy (HYPRE_Solver solver)

- HYPRE_Int HYPRE_ParCSRFlexGMRESSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRFlexGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRFlexGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRFlexGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRFlexGMRESGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)
- HYPRE_Int HYPRE_ParCSRFlexGMRESSetModifyPC (HYPRE_Solver solver, HYPRE_PtrToModifyPCFcn modify_pc)

ParCSR LGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{LGMRES Solver}.

- HYPRE_Int HYPRE_ParCSRLGMRESCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRLGMRESDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRLGMRESSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRLGMRESSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRLGMRESSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_ParCSRLGMRESSetAugDim (HYPRE_Solver solver, HYPRE_Int aug_dim)
- HYPRE_Int HYPRE_ParCSRLGMRESSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRLGMRESSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_ParCSRLGMRESSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRLGMRESSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRLGMRESSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRLGMRESGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRLGMRESSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRLGMRESSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRLGMRESGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRLGMRESGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRLGMRESGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)

ParCSR BiCGSTAB Solver

These routines should be used in conjunction with the generic interface in \Ref{BiCGSTAB Solver}.

- HYPRE_Int HYPRE_ParCSRBiCGSTABCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRBiCGSTABDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)

- HYPRE_Int HYPRE_ParCSRBiCGSTABSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real a_tol)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRBiCGSTABGetPrecond (HYPRE_Solver solver, HYPRE_Solver ∗precond_data)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRBiCGSTABSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRBiCGSTABGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_ParCSRBiCGSTABGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRBiCGSTABGetResidual (HYPRE_Solver solver, HYPRE_ParVector ∗residual)

ParCSR Hybrid Solver

- HYPRE_Int HYPRE_ParCSRHybridCreate (HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRHybridDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRHybridSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRHybridSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRHybridSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRHybridSetAbsoluteTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRHybridSetConvergenceTol (HYPRE_Solver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_ParCSRHybridSetDSCGMaxIter (HYPRE_Solver solver, HYPRE_Int dscg_max_its)
- HYPRE_Int HYPRE_ParCSRHybridSetPCGMaxIter (HYPRE_Solver solver, HYPRE_Int pcg_max_its)
- HYPRE_Int HYPRE_ParCSRHybridSetSetupType (HYPRE_Solver solver, HYPRE_Int setup_type)
- HYPRE_Int HYPRE_ParCSRHybridSetSolverType (HYPRE_Solver solver, HYPRE_Int solver_type)
- HYPRE_Int HYPRE_ParCSRHybridSetRecomputeResidual (HYPRE_Solver solver, HYPRE_Int recompute_residual)
- HYPRE_Int HYPRE_ParCSRHybridGetRecomputeResidual (HYPRE_Solver solver, HYPRE_Int ∗recompute_residual)
- HYPRE_Int HYPRE_ParCSRHybridSetRecomputeResidualP (HYPRE_Solver solver, HYPRE_Int recompute_residual_p)
- HYPRE_Int HYPRE_ParCSRHybridGetRecomputeResidualP (HYPRE_Solver solver, HYPRE_Int ∗recompute_residual_p)
- HYPRE_Int HYPRE_ParCSRHybridSetKDim (HYPRE_Solver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_ParCSRHybridSetTwoNorm (HYPRE_Solver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_ParCSRHybridSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_ParCSRHybridSetRelChange (HYPRE_Solver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_ParCSRHybridSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRHybridSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRHybridSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_ParCSRHybridSetStrongThreshold (HYPRE_Solver solver, HYPRE_Real strong_threshold)
- HYPRE_Int HYPRE_ParCSRHybridSetMaxRowSum (HYPRE_Solver solver, HYPRE_Real max_row_sum)
- HYPRE_Int HYPRE_ParCSRHybridSetTruncFactor (HYPRE_Solver solver, HYPRE_Real trunc_factor)
- HYPRE_Int HYPRE_ParCSRHybridSetPMaxElmts (HYPRE_Solver solver, HYPRE_Int P_max_elmts)
- HYPRE_Int HYPRE_ParCSRHybridSetMaxLevels (HYPRE_Solver solver, HYPRE_Int max_levels)
- HYPRE_Int HYPRE_ParCSRHybridSetMeasureType (HYPRE_Solver solver, HYPRE_Int measure_type)
- HYPRE_Int HYPRE_ParCSRHybridSetCoarsenType (HYPRE_Solver solver, HYPRE_Int coarsen_type)
- HYPRE_Int HYPRE_ParCSRHybridSetInterpType (HYPRE_Solver solver, HYPRE_Int interp_type)

- HYPRE_Int HYPRE_ParCSRHybridSetCycleType (HYPRE_Solver solver, HYPRE_Int cycle_type)
- HYPRE_Int HYPRE_ParCSRHybridSetGridRelaxType (HYPRE_Solver solver, HYPRE_Int ∗grid_relax_type)
- HYPRE_Int HYPRE_ParCSRHybridSetGridRelaxPoints (HYPRE_Solver solver, HYPRE_Int ∗∗grid_relax_↩ points)
- HYPRE_Int HYPRE_ParCSRHybridSetNumSweeps (HYPRE_Solver solver, HYPRE_Int num_sweeps)
- HYPRE_Int HYPRE_ParCSRHybridSetCycleNumSweeps (HYPRE_Solver solver, HYPRE_Int num_sweeps, HYPRE_Int k)
- HYPRE_Int HYPRE_ParCSRHybridSetRelaxType (HYPRE_Solver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_ParCSRHybridSetCycleRelaxType (HYPRE_Solver solver, HYPRE_Int relax_type, HYPRE_Int k)
- HYPRE_Int HYPRE_ParCSRHybridSetRelaxOrder (HYPRE_Solver solver, HYPRE_Int relax_order)
- HYPRE_Int HYPRE_ParCSRHybridSetRelaxWt (HYPRE_Solver solver, HYPRE_Real relax_wt)
- HYPRE_Int HYPRE_ParCSRHybridSetLevelRelaxWt (HYPRE_Solver solver, HYPRE_Real relax_wt, HYPRE_Int level)
- HYPRE_Int HYPRE_ParCSRHybridSetOuterWt (HYPRE_Solver solver, HYPRE_Real outer_wt)
- HYPRE_Int HYPRE_ParCSRHybridSetLevelOuterWt (HYPRE_Solver solver, HYPRE_Real outer_wt, HYPRE_Int level)
- HYPRE_Int HYPRE_ParCSRHybridSetMaxCoarseSize (HYPRE_Solver solver, HYPRE_Int max_coarse_↩ size)
- HYPRE_Int HYPRE_ParCSRHybridSetMinCoarseSize (HYPRE_Solver solver, HYPRE_Int min_coarse_size)
- HYPRE_Int HYPRE_ParCSRHybridSetSeqThreshold (HYPRE_Solver solver, HYPRE_Int seq_threshold)
- HYPRE_Int HYPRE_ParCSRHybridSetRelaxWeight (HYPRE_Solver solver, HYPRE_Real ∗relax_weight)
- HYPRE_Int HYPRE_ParCSRHybridSetOmega (HYPRE_Solver solver, HYPRE_Real ∗omega)
- HYPRE_Int HYPRE_ParCSRHybridSetAggNumLevels (HYPRE_Solver solver, HYPRE_Int agg_num_levels)
- HYPRE_Int HYPRE_ParCSRHybridSetNumPaths (HYPRE_Solver solver, HYPRE_Int num_paths)
- HYPRE_Int HYPRE_ParCSRHybridSetNumFunctions (HYPRE_Solver solver, HYPRE_Int num_functions)
- HYPRE_Int HYPRE_ParCSRHybridSetDofFunc (HYPRE_Solver solver, HYPRE_Int ∗dof_func)
- HYPRE_Int HYPRE_ParCSRHybridSetNodal (HYPRE_Solver solver, HYPRE_Int nodal)
- HYPRE_Int HYPRE_ParCSRHybridSetKeepTranspose (HYPRE_Solver solver, HYPRE_Int keepT)
- HYPRE_Int HYPRE_ParCSRHybridSetNonGalerkinTol (HYPRE_Solver solver, HYPRE_Int num_levels, HYPRE_Real ∗nongalerkin_tol)
- HYPRE_Int HYPRE_ParCSRHybridGetNumIterations (HYPRE_Solver solver, HYPRE_Int ∗num_its)
- HYPRE_Int HYPRE_ParCSRHybridGetDSCGNumIterations (HYPRE_Solver solver, HYPRE_Int ∗dscg_↩ num_its)
- HYPRE_Int HYPRE_ParCSRHybridGetPCGNumIterations (HYPRE_Solver solver, HYPRE_Int ∗pcg_num_its)
- HYPRE_Int HYPRE_ParCSRHybridGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_ParCSRHybridSetNumGridSweeps (HYPRE_Solver solver, HYPRE_Int ∗num_grid_↩ sweeps)
- HYPRE_Int HYPRE_SchwarzCreate (HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_SchwarzDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_SchwarzSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_SchwarzSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_SchwarzSetVariant (HYPRE_Solver solver, HYPRE_Int variant)
- HYPRE_Int HYPRE_SchwarzSetOverlap (HYPRE_Solver solver, HYPRE_Int overlap)
- HYPRE_Int HYPRE_SchwarzSetDomainType (HYPRE_Solver solver, HYPRE_Int domain_type)
- HYPRE_Int HYPRE_SchwarzSetRelaxWeight (HYPRE_Solver solver, HYPRE_Real relax_weight)
- HYPRE_Int HYPRE_SchwarzSetDomainStructure (HYPRE_Solver solver, HYPRE_CSRMatrix domain_↩ structure)
- HYPRE_Int HYPRE_SchwarzSetNumFunctions (HYPRE_Solver solver, HYPRE_Int num_functions)
- HYPRE_Int HYPRE_SchwarzSetDofFunc (HYPRE_Solver solver, HYPRE_Int ∗dof_func)
- HYPRE_Int HYPRE_SchwarzSetNonSymm (HYPRE_Solver solver, HYPRE_Int use_nonsymm)
- HYPRE_Int HYPRE_ParCSRCGNRCreate (MPI_Comm comm, HYPRE_Solver ∗solver)
- HYPRE_Int HYPRE_ParCSRCGNRDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_ParCSRCGNRSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_Par↩ Vector b, HYPRE_ParVector x)

- HYPRE_Int HYPRE_ParCSRCGNRSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_ParCSRCGNRSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_ParCSRCGNRSetMinIter (HYPRE_Solver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_ParCSRCGNRSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_ParCSRCGNRSetStopCrit (HYPRE_Solver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_ParCSRCGNRSetPrecond (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn precond, HYPRE_PtrToParSolverFcn precondT, HYPRE_PtrToParSolverFcn precond_setup, HYPRE_Solver precond_solver)
- HYPRE_Int HYPRE_ParCSRCGNRGetPrecond (HYPRE_Solver solver, HYPRE_Solver *precond_data)
- HYPRE_Int HYPRE_ParCSRCGNRSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_ParCSRCGNRGetNumIterations (HYPRE_Solver solver, HYPRE_Int *num_iterations)
- HYPRE_Int HYPRE_ParCSRCGNRGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real *norm)

ParCSR MGR Solver

Parallel multigrid reduction solver and preconditioner. This solver or preconditioner is designed with systems of PDEs in mind. However, it can also be used for scalar linear systems, particularly for problems where the user can exploit information from the physics of the problem. In this way, the MGR solver could potentially be used as a foundation for a physics-based preconditioner.

- HYPRE_Int HYPRE_MGRCreate (HYPRE_Solver *solver)
- HYPRE_Int HYPRE_MGRDestroy (HYPRE_Solver solver)
- HYPRE_Int HYPRE_MGRSetup (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_MGRSolve (HYPRE_Solver solver, HYPRE_ParCSRMatrix A, HYPRE_ParVector b, HYPRE_ParVector x)
- HYPRE_Int HYPRE_MGRSetCpointsByBlock (HYPRE_Solver solver, HYPRE_Int block_size, HYPRE_Int max_num_levels, HYPRE_Int *num_block_coarse_points, HYPRE_Int **block_coarse_indexes)
- HYPRE_Int HYPRE_MGRSetNonCpointsToFpoints (HYPRE_Solver solver, HYPRE_Int nonCptToFptFlag)
- HYPRE_Int HYPRE_MGRSetMaxCoarseLevels (HYPRE_Solver solver, HYPRE_Int maxlev)
- HYPRE_Int HYPRE_MGRSetBlockSize (HYPRE_Solver solver, HYPRE_Int bsize)
- HYPRE_Int HYPRE_MGRSetReservedCoarseNodes (HYPRE_Solver solver, HYPRE_Int reserved_coarse_size, HYPRE_Int *reserved_coarse_nodes)
- HYPRE_Int HYPRE_MGRSetRelaxType (HYPRE_Solver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_MGRSetFRelaxMethod (HYPRE_Solver solver, HYPRE_Int relax_method)
- HYPRE_Int HYPRE_MGRSetRestrictType (HYPRE_Solver solver, HYPRE_Int restrict_type)
- HYPRE_Int HYPRE_MGRSetNumRestrictSweeps (HYPRE_Solver solver, HYPRE_Int nsweeps)
- HYPRE_Int HYPRE_MGRSetInterpType (HYPRE_Solver solver, HYPRE_Int interp_type)
- HYPRE_Int HYPRE_MGRSetNumRelaxSweeps (HYPRE_Solver solver, HYPRE_Int nsweeps)
- HYPRE_Int HYPRE_MGRSetNumInterpSweeps (HYPRE_Solver solver, HYPRE_Int nsweeps)
- HYPRE_Int HYPRE_MGRSetCoarseSolver (HYPRE_Solver solver, HYPRE_PtrToParSolverFcn coarse_grid_solver_solve, HYPRE_PtrToParSolverFcn coarse_grid_solver_setup, HYPRE_Solver coarse_grid_solver)
- HYPRE_Int HYPRE_MGRSetPrintLevel (HYPRE_Solver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_MGRSetLogging (HYPRE_Solver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_MGRSetMaxIter (HYPRE_Solver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_MGRSetTol (HYPRE_Solver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_MGRSetMaxGlobalsmoothIters (HYPRE_Solver solver, HYPRE_Int smooth_iter)
- HYPRE_Int HYPRE_MGRSetGlobalsmoothType (HYPRE_Solver solver, HYPRE_Int smooth_type)
- HYPRE_Int HYPRE_MGRGetNumIterations (HYPRE_Solver solver, HYPRE_Int *num_iterations)
- HYPRE_Int HYPRE_MGRGetFinalRelativeResidualNorm (HYPRE_Solver solver, HYPRE_Real *res_norm)
- HYPRE_ParCSRMatrix GenerateLaplacian (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Real *value)
- HYPRE_ParCSRMatrix GenerateLaplacian27pt (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Real *value)

- HYPRE_ParCSRMatrix GenerateLaplacian9pt (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int p, HYPRE_Int q, HYPRE_Real ∗value)
- HYPRE_ParCSRMatrix GenerateDifConv (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Real ∗value)
- HYPRE_ParCSRMatrix GenerateRotate7pt (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int p, HYPRE_Int q, HYPRE_Real alpha, HYPRE_Real eps)
- HYPRE_ParCSRMatrix GenerateVarDifConv (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Real eps, HYPRE_ParVector ∗rhs_ptr)
- HYPRE_ParCSRMatrix GenerateRSVarDifConv (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Real eps, HYPRE_ParVector ∗rhs_ptr, HYPRE_Int type)
- float ∗ GenerateCoordinates (MPI_Comm comm, HYPRE_BigInt nx, HYPRE_BigInt ny, HYPRE_BigInt nz, HYPRE_Int P, HYPRE_Int Q, HYPRE_Int R, HYPRE_Int p, HYPRE_Int q, HYPRE_Int r, HYPRE_Int coorddim)
- HYPRE_Int HYPRE_BoomerAMGSetPostInterpType (HYPRE_Solver solver, HYPRE_Int post_interp_type)
- HYPRE_Int HYPRE_BoomerAMGSetJacobiTruncThreshold (HYPRE_Solver solver, HYPRE_Real jacobi_↩ trunc_threshold)
- HYPRE_Int HYPRE_BoomerAMGSetNumCRRelaxSteps (HYPRE_Solver solver, HYPRE_Int num_CR_↩ relax_steps)
- HYPRE_Int HYPRE_BoomerAMGSetCRRate (HYPRE_Solver solver, HYPRE_Real CR_rate)
- HYPRE_Int HYPRE_BoomerAMGSetCRStrongTh (HYPRE_Solver solver, HYPRE_Real CR_strong_th)
- HYPRE_Int HYPRE_BoomerAMGSetCRUseCG (HYPRE_Solver solver, HYPRE_Int CR_use_CG)
- HYPRE_Int HYPRE_BoomerAMGSetISType (HYPRE_Solver solver, HYPRE_Int IS_type)

ParCSR LOBPCG Eigensolver

These routines should be used in conjunction with the generic interface in \Ref{LOBPCG Eigensolver}.

- HYPRE_Int HYPRE_ParCSRSetupInterpreter (mv_InterfaceInterpreter ∗i)
- HYPRE_Int HYPRE_ParCSRSetupMatvec (HYPRE_MatvecFunctions ∗mv)
- HYPRE_Int hypre_ParCSRMultiVectorPrint (void ∗x_, const char ∗fileName)
- void ∗ hypre_ParCSRMultiVectorRead (MPI_Comm comm, void ∗ii_, const char ∗fileName)

ParCSR Solvers

- #define HYPRE_SOLVER_STRUCT
- #define HYPRE_MODIFYPC
- typedef struct hypre_Solver_struct ∗ HYPRE_Solver
- typedef HYPRE_Int(∗ HYPRE_PtrToParSolverFcn) (HYPRE_Solver, HYPRE_ParCSRMatrix, HYPRE_ParVector, HYPRE_ParVector)
- typedef HYPRE_Int(∗ HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)

2.4.1 Macro Definition Documentation

2.4.1.1 HYPRE_MODIFYPC

```
#define HYPRE_MODIFYPC
```

2.4.1.2 HYPRE_SOLVER_STRUCT

```
#define HYPRE_SOLVER_STRUCT
```

The solver object.

2.4.2 Typedef Documentation

2.4.2.1 HYPRE_PtrToModifyPCFcn

```
typedef HYPRE_Int(* HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)
```

2.4.2.2 HYPRE_PtrToParSolverFcn

```
typedef HYPRE_Int(* HYPRE_PtrToParSolverFcn) (HYPRE_Solver, HYPRE_ParCSRMatrix, HYPRE_ParVector,
HYPRE_ParVector)
```

2.4.2.3 HYPRE_Solver

```
typedef struct hypre_Solver_struct* HYPRE_Solver
```

2.4.3 Function Documentation

2.4.3.1 GenerateCoordinates()

```
float* GenerateCoordinates (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Int coorddim )
```

2.4.3.2 GenerateDifConv()

```
HYPRE_ParCSRMatrix GenerateDifConv (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Real * value )
```

2.4.3.3 GenerateLaplacian()

```
HYPRE_ParCSRMatrix GenerateLaplacian (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Real * value )
```

2.4.3.4 GenerateLaplacian27pt()

```
HYPRE_ParCSRMatrix GenerateLaplacian27pt (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Real * value )
```

2.4.3.5 GenerateLaplacian9pt()

```
HYPRE_ParCSRMatrix GenerateLaplacian9pt (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Real * value )
```

2.4.3.6 GenerateRotate7pt()

```
HYPRE_ParCSRMatrix GenerateRotate7pt (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Real alpha,
            HYPRE_Real eps )
```

2.4.3.7 GenerateRSVarDifConv()

```
HYPRE_ParCSRMatrix GenerateRSVarDifConv (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Real eps,
            HYPRE_ParVector * rhs_ptr,
            HYPRE_Int type )
```

2.4.3.8 GenerateVarDifConv()

```
HYPRE_ParCSRMatrix GenerateVarDifConv (
            MPI_Comm comm,
            HYPRE_BigInt nx,
            HYPRE_BigInt ny,
            HYPRE_BigInt nz,
            HYPRE_Int P,
            HYPRE_Int Q,
            HYPRE_Int R,
            HYPRE_Int p,
            HYPRE_Int q,
            HYPRE_Int r,
            HYPRE_Real eps,
            HYPRE_ParVector * rhs_ptr )
```

2.4.3.9 HYPRE_ADSCreate()

```
HYPRE_Int HYPRE_ADSCreate (
            HYPRE_Solver * solver )
```

Create an ADS solver object.

2.4.3.10 HYPRE_ADSDestroy()

```
HYPRE_Int HYPRE_ADSDestroy (
            HYPRE_Solver solver )
```

Destroy an ADS solver object.

2.4.3.11 HYPRE_ADSGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ADSGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * rel_resid_norm )
```

Returns the norm of the final relative residual.

2.4.3.12 HYPRE_ADSGetNumIterations()

```
HYPRE_Int HYPRE_ADSGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Returns the number of iterations taken.

2.4.3.13 HYPRE_ADSSetAMGOptions()

```
HYPRE_Int HYPRE_ADSSetAMGOptions (
            HYPRE_Solver solver,
            HYPRE_Int coarsen_type,
            HYPRE_Int agg_levels,
            HYPRE_Int relax_type,
            HYPRE_Real strength_threshold,
            HYPRE_Int interp_type,
            HYPRE_Int Pmax )
```

(Optional) Sets AMG parameters for B_Π. The defaults are 10, 1, 3, 0.25, 0, 0. See the user's manual for more details.

2.4.3.14 HYPRE_ADSSetAMSOptions()

```
HYPRE_Int HYPRE_ADSSetAMSOptions (
            HYPRE_Solver solver,
            HYPRE_Int cycle_type,
            HYPRE_Int coarsen_type,
            HYPRE_Int agg_levels,
            HYPRE_Int relax_type,
            HYPRE_Real strength_threshold,
            HYPRE_Int interp_type,
            HYPRE_Int Pmax )
```

(Optional) Sets AMS parameters for B_C. The defaults are 11, 10, 1, 3, 0.25, 0, 0. Note that cycle↩
_type should be greater than 10, unless the high-order interface of HYPRE_ADSSetInterpolations is being used! See the user's manual for more details.

2.4.3.15 HYPRE_ADSSetChebySmoothingOptions()

```
HYPRE_Int HYPRE_ADSSetChebySmoothingOptions (
            HYPRE_Solver solver,
            HYPRE_Int cheby_order,
            HYPRE_Int cheby_fraction )
```

(Optional) Sets parameters for Chebyshev relaxation. The defaults are 2, 0.3.

2.4.3.16 HYPRE_ADSSetCoordinateVectors()

```
HYPRE_Int HYPRE_ADSSetCoordinateVectors (
            HYPRE_Solver solver,
            HYPRE_ParVector x,
            HYPRE_ParVector y,
            HYPRE_ParVector z )
```

Sets the x, y and z coordinates of the vertices in the mesh. This function should be called before HYPRE_ADSSetup()!

2.4.3.17 HYPRE_ADSSetCycleType()

```
HYPRE_Int HYPRE_ADSSetCycleType (
            HYPRE_Solver solver,
            HYPRE_Int cycle_type )
```

(Optional) Choose which auxiliary-space solver to use. Possible values are:

\begin{tabular}{|c|l|} \hline 1 & 3-level multiplicative solver (01210) \ 2 & 3-level additive solver (0+1+2) \ 3 & 3-level multiplicative solver (02120) \ 4 & 3-level additive solver (010+2) \ 5 & 3-level multiplicative solver (0102010) \ 6 & 3-level additive solver (1+020) \ 7 & 3-level multiplicative solver (0201020) \ 8 & 3-level additive solver (0(1+2)0) \ 11 & 5-level multiplicative solver (013454310) \ 12 & 5-level additive solver (0+1+3+4+5) \ 13 & 5-level multiplicative solver (034515430) \ 14 & 5-level additive solver (01(3+4+5)10) \ \hline \end{tabular}

The default is 1. See the user's manual for more details.

2.4.3.18 HYPRE_ADSSetDiscreteCurl()

```
HYPRE_Int HYPRE_ADSSetDiscreteCurl (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix C )
```

Sets the discrete curl matrix C. This function should be called before HYPRE_ADSSetup()!

2.4.3.19 HYPRE_ADSSetDiscreteGradient()

```
HYPRE_Int HYPRE_ADSSetDiscreteGradient (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix G )
```

Sets the discrete gradient matrix G. This function should be called before HYPRE_ADSSetup()!

2.4.3.20 HYPRE_ADSSetInterpolations()

```
HYPRE_Int HYPRE_ADSSetInterpolations (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix RT_Pi,
            HYPRE_ParCSRMatrix RT_Pix,
            HYPRE_ParCSRMatrix RT_Piy,
            HYPRE_ParCSRMatrix RT_Piz,
            HYPRE_ParCSRMatrix ND_Pi,
            HYPRE_ParCSRMatrix ND_Pix,
            HYPRE_ParCSRMatrix ND_Piy,
            HYPRE_ParCSRMatrix ND_Piz )
```

$ – only one of them needs to be specified, and the availability of each enables different AMS cycle type options.

2.4.3.21 HYPRE_ADSSetMaxIter()

```
HYPRE_Int HYPRE_ADSSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int maxit )
```

(Optional) Sets maximum number of iterations, if ADS is used as a solver. To use ADS as a preconditioner, set the maximum number of iterations to 1. The default is 20.

2.4.3.22 HYPRE_ADSSetPrintLevel()

```
HYPRE_Int HYPRE_ADSSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

(Optional) Control how much information is printed during the solution iterations. The default is 1 (print residual norm at each step).

2.4.3.23 HYPRE_ADSSetSmoothingOptions()

```
HYPRE_Int HYPRE_ADSSetSmoothingOptions (
            HYPRE_Solver solver,
            HYPRE_Int relax_type,
            HYPRE_Int relax_times,
            HYPRE_Real relax_weight,
            HYPRE_Real omega )
```

(Optional) Sets relaxation parameters for A. The defaults are 2, 1, 1.0, 1.0.

The available options for relax_type are:

\begin{tabular}{|c|l|} \hline 1 & ℓ_1-scaled Jacobi \ 2 & ℓ_1-scaled block symmetric Gauss-Seidel/SSOR \ 3 & Kaczmarz \ 4 & truncated version of ℓ_1-scaled block symmetric Gauss-Seidel/SSOR \ 16 & Chebyshev \ \hline \end{tabular}

2.4.3.24 HYPRE_ADSSetTol()

```
HYPRE_Int HYPRE_ADSSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance, if ADS is used as a solver. When using ADS as a preconditioner, set the tolerance to 0.0. The default is 10^{-6}.

2.4.3.25 HYPRE_ADSSetup()

```
HYPRE_Int HYPRE_ADSSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Set up the ADS solver or preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] object to be set up.
A	[IN] ParCSR matrix used to construct the solver/preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.26 HYPRE_ADSSolve()

```
HYPRE_Int HYPRE_ADSSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve the system or apply ADS as a preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix, matrix of the linear system to be solved
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.27 HYPRE_AMSConstructDiscreteGradient()

```
HYPRE_Int HYPRE_AMSConstructDiscreteGradient (
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector x_coord,
            HYPRE_BigInt * edge_vertex,
            HYPRE_Int edge_orientation,
            HYPRE_ParCSRMatrix * G )
```

Construct and return the lowest-order discrete gradient matrix G using some edge and vertex information. We assume that edge_vertex lists the edge vertices consecutively, and that the orientation of all edges is consistent.

If edge_orientation = 1, the edges are already oriented.

If edge_orientation = 2, the orientation of edge i depends only on the sign of edge_vertex[2∗i+1] - edge_vertex[2∗i].

2.4.3.28 HYPRE_AMSCreate()

```
HYPRE_Int HYPRE_AMSCreate (
            HYPRE_Solver * solver )
```

Create an AMS solver object.

2.4.3.29 HYPRE_AMSDestroy()

```
HYPRE_Int HYPRE_AMSDestroy (
            HYPRE_Solver solver )
```

Destroy an AMS solver object.

2.4.3.30 HYPRE_AMSGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_AMSGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * rel_resid_norm )
```

Returns the norm of the final relative residual.

2.4.3.31 HYPRE_AMSGetNumIterations()

```
HYPRE_Int HYPRE_AMSGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Returns the number of iterations taken.

2.4.3.32 HYPRE_AMSProjectOutGradients()

```
HYPRE_Int HYPRE_AMSProjectOutGradients (
            HYPRE_Solver solver,
            HYPRE_ParVector x )
```

For problems with zero-conductivity regions, project the vector onto the compatible subspace: $x = (I - G_0 (G_0^t G_0)^{-1} G_0^T) x$, where G_0 is the discrete gradient restricted to the interior nodes of the regions with zero conductivity. This ensures that x is orthogonal to the gradients in the range of G_0.

This function is typically called after the solution iteration is complete, in order to facilitate the visualization of the computed field. Without it the values in the zero-conductivity regions contain kernel components.

2.4.3.33 HYPRE_AMSSetAlphaAMGCoarseRelaxType()

```
HYPRE_Int HYPRE_AMSSetAlphaAMGCoarseRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int alpha_coarse_relax_type )
```

(Optional) Sets the coarsest level relaxation in the AMG solver for B_Π. The default is 8 (l1-GS). Use 9, 19, 29 or 99 for a direct solver.

2.4.3.34 HYPRE_AMSSetAlphaAMGOptions()

```
HYPRE_Int HYPRE_AMSSetAlphaAMGOptions (
            HYPRE_Solver solver,
            HYPRE_Int alpha_coarsen_type,
            HYPRE_Int alpha_agg_levels,
            HYPRE_Int alpha_relax_type,
            HYPRE_Real alpha_strength_threshold,
            HYPRE_Int alpha_interp_type,
            HYPRE_Int alpha_Pmax )
```

(Optional) Sets AMG parameters for B_Π. The defaults are 10, 1, 3, 0.25, 0, 0. See the user's manual for more details.

2.4.3.35 HYPRE_AMSSetAlphaPoissonMatrix()

```
HYPRE_Int HYPRE_AMSSetAlphaPoissonMatrix (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A_alpha )
```

(Optional) Sets the matrix A_α corresponding to the Poisson problem with coefficient α (the curl-curl term coefficient in the Maxwell problem).

If this function is called, the coarse space solver on the range of Π^T is a block-diagonal version of A_Π. If this function is not called, the coarse space solver on the range of Π^T is constructed as $\Pi^T A \Pi$ in HYPRE_AMSSetup(). See the user's manual for more details.

2.4.3.36 HYPRE_AMSSetBetaAMGCoarseRelaxType()

```
HYPRE_Int HYPRE_AMSSetBetaAMGCoarseRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int beta_coarse_relax_type )
```

(Optional) Sets the coarsest level relaxation in the AMG solver for B_G. The default is 8 (l1-GS). Use 9, 19, 29 or 99 for a direct solver.

2.4.3.37 HYPRE_AMSSetBetaAMGOptions()

```
HYPRE_Int HYPRE_AMSSetBetaAMGOptions (
            HYPRE_Solver solver,
            HYPRE_Int beta_coarsen_type,
            HYPRE_Int beta_agg_levels,
            HYPRE_Int beta_relax_type,
            HYPRE_Real beta_strength_threshold,
            HYPRE_Int beta_interp_type,
            HYPRE_Int beta_Pmax )
```

(Optional) Sets AMG parameters for B_G. The defaults are 10, 1, 3, 0.25, 0, 0. See the user's manual for more details.

2.4.3.38 HYPRE_AMSSetBetaPoissonMatrix()

```
HYPRE_Int HYPRE_AMSSetBetaPoissonMatrix (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A_beta )
```

(Optional) Sets the matrix A_β corresponding to the Poisson problem with coefficient β (the mass term coefficient in the Maxwell problem).

If not given, the Poisson matrix will be computed in HYPRE_AMSSetup(). If the given matrix is NULL, we assume that β is identically 0 and use two-level (instead of three-level) methods. See the user's manual for more details.

2.4.3.39 HYPRE_AMSSetCoordinateVectors()

```
HYPRE_Int HYPRE_AMSSetCoordinateVectors (
            HYPRE_Solver solver,
            HYPRE_ParVector x,
            HYPRE_ParVector y,
            HYPRE_ParVector z )
```

Sets the x, y and z coordinates of the vertices in the mesh.

Either HYPRE_AMSSetCoordinateVectors() or HYPRE_AMSSetEdgeConstantVectors() should be called before HYPRE_AMSSetup()!

2.4.3.40 HYPRE_AMSSetCycleType()

```
HYPRE_Int HYPRE_AMSSetCycleType (
            HYPRE_Solver solver,
            HYPRE_Int cycle_type )
```

(Optional) Choose which three-level solver to use. Possible values are:

\begin{tabular}{|c|l|} \hline 1 & 3-level multiplicative solver (01210) \ 2 & 3-level additive solver (0+1+2) \ 3 & 3-level multiplicative solver (02120) \ 4 & 3-level additive solver (010+2) \ 5 & 3-level multiplicative solver (0102010) \ 6 & 3-level additive solver (1+020) \ 7 & 3-level multiplicative solver (0201020) \ 8 & 3-level additive solver (0(1+2)0) \ 11 & 5-level multiplicative solver (013454310) \ 12 & 5-level additive solver (0+1+3+4+5) \ 13 & 5-level multiplicative solver (034515430) \ 14 & 5-level additive solver (01(3+4+5)10) \ \hline \end{tabular}

The default is 1. See the user's manual for more details.

2.4.3.41 HYPRE_AMSSetDimension()

```
HYPRE_Int HYPRE_AMSSetDimension (
            HYPRE_Solver solver,
            HYPRE_Int dim )
```

(Optional) Sets the problem dimension (2 or 3). The default is 3.

2.4.3.42 HYPRE_AMSSetDiscreteGradient()

```
HYPRE_Int HYPRE_AMSSetDiscreteGradient (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix G )
```

Sets the discrete gradient matrix G. This function should be called before HYPRE_AMSSetup()!

2.4.3.43 HYPRE_AMSSetEdgeConstantVectors()

```
HYPRE_Int HYPRE_AMSSetEdgeConstantVectors (
            HYPRE_Solver solver,
            HYPRE_ParVector Gx,
            HYPRE_ParVector Gy,
            HYPRE_ParVector Gz )
```

Sets the vectors Gx, Gy and Gz which give the representations of the constant vector fields $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ in the edge element basis.

Either HYPRE_AMSSetCoordinateVectors() or HYPRE_AMSSetEdgeConstantVectors() should be called before HYPRE_AMSSetup()!

2.4.3.44 HYPRE_AMSSetInteriorNodes()

```
HYPRE_Int HYPRE_AMSSetInteriorNodes (
            HYPRE_Solver solver,
            HYPRE_ParVector interior_nodes )
```

(Optional) Set the list of nodes which are interior to a zero-conductivity region. This way, a more robust solver is constructed, that can be iterated to lower tolerance levels. A node is interior if its entry in the array is 1.0. This function should be called before HYPRE_AMSSetup()!

2.4.3.45 HYPRE_AMSSetInterpolations()

```
HYPRE_Int HYPRE_AMSSetInterpolations (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix Pi,
            HYPRE_ParCSRMatrix Pix,
            HYPRE_ParCSRMatrix Piy,
            HYPRE_ParCSRMatrix Piz )
```

$ needs to be specified (though it is OK to provide both). If Pix is NULL, then scalar Π-based AMS cycles, i.e. those with cycle_type $>$ 10, will be unavailable. Similarly, AMS cycles based on monolithic Π (cycle_type $<$ 10) require that Pi is not NULL.

2.4.3.46 HYPRE_AMSSetMaxIter()

```
HYPRE_Int HYPRE_AMSSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int maxit )
```

(Optional) Sets maximum number of iterations, if AMS is used as a solver. To use AMS as a preconditioner, set the maximum number of iterations to 1. The default is 20.

2.4.3.47 HYPRE_AMSSetPrintLevel()

```
HYPRE_Int HYPRE_AMSSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

(Optional) Control how much information is printed during the solution iterations. The default is 1 (print residual norm at each step).

2.4.3.48 HYPRE_AMSSetProjectionFrequency()

```
HYPRE_Int HYPRE_AMSSetProjectionFrequency (
            HYPRE_Solver solver,
            HYPRE_Int projection_frequency )
```

(Optional) Set the frequency at which a projection onto the compatible subspace for problems with zero-conductivity regions is performed. The default value is 5.

2.4.3.49 HYPRE_AMSSetSmoothingOptions()

```
HYPRE_Int HYPRE_AMSSetSmoothingOptions (
            HYPRE_Solver solver,
            HYPRE_Int relax_type,
            HYPRE_Int relax_times,
            HYPRE_Real relax_weight,
            HYPRE_Real omega )
```

(Optional) Sets relaxation parameters for A. The defaults are 2, 1, 1.0, 1.0.

The available options for relax_type are:

\begin{tabular}{|c|l|} \hline 1 & ℓ_1-scaled Jacobi \ 2 & ℓ_1-scaled block symmetric Gauss-Seidel/SSOR \ 3 & Kaczmarz \ 4 & truncated version of ℓ_1-scaled block symmetric Gauss-Seidel/SSOR \ 16 & Chebyshev \ \hline \end{tabular}

2.4.3.50 HYPRE_AMSSetTol()

```
HYPRE_Int HYPRE_AMSSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance, if AMS is used as a solver. When using AMS as a preconditioner, set the tolerance to 0.0. The default is 10^{-6}.

2.4.3.51 HYPRE_AMSSetup()

```
HYPRE_Int HYPRE_AMSSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Set up the AMS solver or preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] object to be set up.
A	[IN] ParCSR matrix used to construct the solver/preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.52 HYPRE_AMSSolve()

```
HYPRE_Int HYPRE_AMSSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve the system or apply AMS as a preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix, matrix of the linear system to be solved
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.53 HYPRE_BoomerAMGCreate()

```
HYPRE_Int HYPRE_BoomerAMGCreate (
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.54 HYPRE_BoomerAMGDestroy()

```
HYPRE_Int HYPRE_BoomerAMGDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.55 HYPRE_BoomerAMGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_BoomerAMGGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * rel_resid_norm )
```

Returns the norm of the final relative residual.

2.4.3.56 HYPRE_BoomerAMGGetNumIterations()

```
HYPRE_Int HYPRE_BoomerAMGGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

Returns the number of iterations taken.

2.4.3.57 HYPRE_BoomerAMGGetResidual()

```
HYPRE_Int HYPRE_BoomerAMGGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

Returns the residual.

2.4.3.58 HYPRE_BoomerAMGInitGridRelaxation()

```
HYPRE_Int HYPRE_BoomerAMGInitGridRelaxation (
            HYPRE_Int ** num_grid_sweeps_ptr,
            HYPRE_Int ** grid_relax_type_ptr,
            HYPRE_Int *** grid_relax_points_ptr,
            HYPRE_Int coarsen_type,
            HYPRE_Real ** relax_weights_ptr,
            HYPRE_Int max_levels )
```

(Optional) This routine will be eliminated in the future.

2.4.3.59 HYPRE_BoomerAMGSetAdditive()

```
HYPRE_Int HYPRE_BoomerAMGSetAdditive (
            HYPRE_Solver solver,
            HYPRE_Int addlvl )
```

(Optional) Defines use of an additive V(1,1)-cycle using the classical additive method starting at level 'addlvl'. The multiplicative approach is used on levels 0, ...'addlvl+1'. 'addlvl' needs to be $>$ -1 for this to have an effect. Can only be used with weighted Jacobi and l1-Jacobi(default).

Can only be used when AMG is used as a preconditioner !!!

2.4.3.60 HYPRE_BoomerAMGSetAddLastLvl()

```
HYPRE_Int HYPRE_BoomerAMGSetAddLastLvl (
            HYPRE_Solver solver,
            HYPRE_Int add_last_lvl )
```

(Optional) Defines last level where additive, mult-additive or simple cycle is used. The multiplicative approach is used on levels $>$ add_last_lvl.

Can only be used when AMG is used as a preconditioner !!!

2.4.3.61 HYPRE_BoomerAMGSetAddRelaxType()

```
HYPRE_Int HYPRE_BoomerAMGSetAddRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int add_rlx_type )
```

(Optional) Defines the relaxation type used in the (mult)additive cycle portion (also affects simple method.) The default is 18 (L1-Jacobi). Currently the only other option allowed is 0 (Jacobi) which should be used in combination with HYPRE_BoomerAMGSetAddRelaxWt.

2.4.3.62 HYPRE_BoomerAMGSetAddRelaxWt()

```
HYPRE_Int HYPRE_BoomerAMGSetAddRelaxWt (
            HYPRE_Solver solver,
            HYPRE_Real add_rlx_wt )
```

(Optional) Defines the relaxation weight used for Jacobi within the (mult)additive or simple cycle portion. The default is 1. The weight only affects the Jacobi method, and has no effect on L1-Jacobi

2.4.3.63 HYPRE_BoomerAMGSetADropTol()

```
HYPRE_Int HYPRE_BoomerAMGSetADropTol (
            HYPRE_Solver solver,
            HYPRE_Real A_drop_tol )
```

(Optional) Defines the drop tolerance for the A-matrices from the 2nd level of AMG. The default is 0.

2.4.3.64 HYPRE_BoomerAMGSetADropType()

```
HYPRE_Int HYPRE_BoomerAMGSetADropType (
            HYPRE_Solver solver,
            HYPRE_Int A_drop_type )
```

2.4.3.65 HYPRE_BoomerAMGSetAggInterpType()

```
HYPRE_Int HYPRE_BoomerAMGSetAggInterpType (
            HYPRE_Solver solver,
            HYPRE_Int agg_interp_type )
```

(Optional) Defines the interpolation used on levels of aggressive coarsening The default is 4, i.e. multipass interpolation. The following options exist:

\begin{tabular}{|c|l|} \hline 1 & 2-stage extended+i interpolation \ 2 & 2-stage standard interpolation \ 3 & 2-stage extended interpolation \ 4 & multipass interpolation \ \hline \end{tabular}

2.4.3.66 HYPRE_BoomerAMGSetAggNumLevels()

```
HYPRE_Int HYPRE_BoomerAMGSetAggNumLevels (
            HYPRE_Solver solver,
            HYPRE_Int agg_num_levels )
```

(Optional) Defines the number of levels of aggressive coarsening. The default is 0, i.e. no aggressive coarsening.

2.4.3.67 HYPRE_BoomerAMGSetAggP12MaxElmts()

```
HYPRE_Int HYPRE_BoomerAMGSetAggP12MaxElmts (
            HYPRE_Solver solver,
            HYPRE_Int agg_P12_max_elmts )
```

(Optional) Defines the maximal number of elements per row for the matrices P1 and P2 which are used to build 2-stage interpolation. The default is 0.

2.4.3.68 HYPRE_BoomerAMGSetAggP12TruncFactor()

```
HYPRE_Int HYPRE_BoomerAMGSetAggP12TruncFactor (
            HYPRE_Solver solver,
            HYPRE_Real agg_P12_trunc_factor )
```

(Optional) Defines the truncation factor for the matrices P1 and P2 which are used to build 2-stage interpolation. The default is 0.

2.4.3.69 HYPRE_BoomerAMGSetAggPMaxElmts()

```
HYPRE_Int HYPRE_BoomerAMGSetAggPMaxElmts (
            HYPRE_Solver solver,
            HYPRE_Int agg_P_max_elmts )
```

(Optional) Defines the maximal number of elements per row for the interpolation used for aggressive coarsening. The default is 0.

2.4.3.70 HYPRE_BoomerAMGSetAggTruncFactor()

```
HYPRE_Int HYPRE_BoomerAMGSetAggTruncFactor (
            HYPRE_Solver solver,
            HYPRE_Real agg_trunc_factor )
```

(Optional) Defines the truncation factor for the interpolation used for aggressive coarsening. The default is 0.

2.4.3.71 HYPRE_BoomerAMGSetCGCIts()

```
HYPRE_Int HYPRE_BoomerAMGSetCGCIts (
            HYPRE_Solver solver,
            HYPRE_Int its )
```

(optional) Defines the number of pathes for CGC-coarsening.

2.4.3.72 HYPRE_BoomerAMGSetChebyEigEst()

```
HYPRE_Int HYPRE_BoomerAMGSetChebyEigEst (
            HYPRE_Solver solver,
            HYPRE_Int eig_est )
```

2.4.3.73 HYPRE_BoomerAMGSetChebyFraction()

```
HYPRE_Int HYPRE_BoomerAMGSetChebyFraction (
            HYPRE_Solver solver,
            HYPRE_Real ratio )
```

(Optional) Fraction of the spectrum to use for the Chebyshev smoother. The default is .3 (i.e., damp on upper 30% of the spectrum).

2.4.3.74 HYPRE_BoomerAMGSetChebyOrder()

```
HYPRE_Int HYPRE_BoomerAMGSetChebyOrder (
            HYPRE_Solver solver,
            HYPRE_Int order )
```

(Optional) Defines the Order for Chebyshev smoother. The default is 2 (valid options are 1-4).

2.4.3.75 HYPRE_BoomerAMGSetChebyScale()

```
HYPRE_Int HYPRE_BoomerAMGSetChebyScale (
            HYPRE_Solver solver,
            HYPRE_Int scale )
```

2.4.3.76 HYPRE_BoomerAMGSetChebyVariant()

```
HYPRE_Int HYPRE_BoomerAMGSetChebyVariant (
            HYPRE_Solver solver,
            HYPRE_Int variant )
```

2.4.3.77 HYPRE_BoomerAMGSetCoarsenType()

```
HYPRE_Int HYPRE_BoomerAMGSetCoarsenType (
            HYPRE_Solver solver,
            HYPRE_Int coarsen_type )
```

(Optional) Defines which parallel coarsening algorithm is used. There are the following options for coarsen_type:

\begin{tabular}{|c|l|} \hline 0 & CLJP-coarsening (a parallel coarsening algorithm using independent sets. \ 1 & classical Ruge-Stueben coarsening on each processor, no boundary treatment (not recommended!) \ 3 & classical Ruge-↩ Stueben coarsening on each processor, followed by a third pass, which adds coarse \ & points on the boundaries \ 6 & Falgout coarsening (uses 1 first, followed by CLJP using the interior coarse points \ & generated by 1 as its first independent set) \ 7 & CLJP-coarsening (using a fixed random vector, for debugging purposes only) \ 8 & PMIS-coarsening (a parallel coarsening algorithm using independent sets, generating \ & lower complexities than CLJP, might also lead to slower convergence) \ 9 & PMIS-coarsening (using a fixed random vector, for debugging purposes only) \ 10 & HMIS-coarsening (uses one pass Ruge-Stueben on each processor independently, followed \ & by PMIS using the interior C-points generated as its first independent set) \ 11 & one-pass Ruge-Stueben coarsening on each processor, no boundary treatment (not recommended!) \ 21 & CGC coarsening by M. Griebel, B. Metsch and A. Schweitzer \ 22 & CGC-E coarsening by M. Griebel, B. Metsch and A.Schweitzer \ \hline \end{tabular}

The default is 10.

2.4.3.78 HYPRE_BoomerAMGSetConvergeType()

```
HYPRE_Int HYPRE_BoomerAMGSetConvergeType (
            HYPRE_Solver solver,
            HYPRE_Int type )
```

(Optional) Set the type convergence checking 0: (default) norm(r)/norm(b), or norm(r) when b == 0 1: nomr(r) / norm(r_0)

2.4.3.79 HYPRE_BoomerAMGSetCoordDim()

```
HYPRE_Int HYPRE_BoomerAMGSetCoordDim (
            HYPRE_Solver solver,
            HYPRE_Int coorddim )
```

2.4.3.80 HYPRE_BoomerAMGSetCoordinates()

```
HYPRE_Int HYPRE_BoomerAMGSetCoordinates (
            HYPRE_Solver solver,
            float * coordinates )
```

2.4.3.81 HYPRE_BoomerAMGSetCpointsToKeep()

```
HYPRE_Int HYPRE_BoomerAMGSetCpointsToKeep (
            HYPRE_Solver solver,
            HYPRE_Int cpt_coarse_level,
            HYPRE_Int num_cpt_coarse,
            HYPRE_Int * cpt_coarse_index )
```

(Optional) Fix C points to be kept till a specified coarse level.

Parameters

solver	[IN] solver or preconditioner
cpt_coarse_level	[IN] coarse level up to which to keep C points
num_cpt_coarse	[IN] number of C points to be kept
cpt_coarse_index	[IN] indexes of C points to be kept

2.4.3.82 HYPRE_BoomerAMGSetCRRate()

```
HYPRE_Int HYPRE_BoomerAMGSetCRRate (
            HYPRE_Solver solver,
            HYPRE_Real CR_rate )
```

2.4.3.83 HYPRE_BoomerAMGSetCRStrongTh()

```
HYPRE_Int HYPRE_BoomerAMGSetCRStrongTh (
            HYPRE_Solver solver,
            HYPRE_Real CR_strong_th )
```

2.4.3.84 HYPRE_BoomerAMGSetCRUseCG()

```
HYPRE_Int HYPRE_BoomerAMGSetCRUseCG (
            HYPRE_Solver solver,
            HYPRE_Int CR_use_CG )
```

2.4.3.85 HYPRE_BoomerAMGSetCycleNumSweeps()

```
HYPRE_Int HYPRE_BoomerAMGSetCycleNumSweeps (
            HYPRE_Solver solver,
            HYPRE_Int num_sweeps,
            HYPRE_Int k )
```

(Optional) Sets the number of sweeps at a specified cycle. There are the following options for k:

\begin{tabular}{|l|l|} \hline the down cycle & if k=1 \ the up cycle & if k=2 \ the coarsest level & if k=3.\ \hline \end{tabular}

2.4.3.86 HYPRE_BoomerAMGSetCycleRelaxType()

```
HYPRE_Int HYPRE_BoomerAMGSetCycleRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int relax_type,
            HYPRE_Int k )
```

(Optional) Defines the smoother at a given cycle. For options of relax_type see description of HYPRE_Boomer↩ AMGSetRelaxType). Options for k are

\begin{tabular}{|l|l|} \hline the down cycle & if k=1 \ the up cycle & if k=2 \ the coarsest level & if k=3. \ \hline \end{tabular}

2.4.3.87 HYPRE_BoomerAMGSetCycleType()

```
HYPRE_Int HYPRE_BoomerAMGSetCycleType (
            HYPRE_Solver solver,
            HYPRE_Int cycle_type )
```

(Optional) Defines the type of cycle. For a V-cycle, set cycle_type to 1, for a W-cycle set cycle_type to 2. The default is 1.

2.4.3.88 HYPRE_BoomerAMGSetDebugFlag()

```
HYPRE_Int HYPRE_BoomerAMGSetDebugFlag (
            HYPRE_Solver solver,
            HYPRE_Int debug_flag )
```

(Optional)

2.4.3.89 HYPRE_BoomerAMGSetDofFunc()

```
HYPRE_Int HYPRE_BoomerAMGSetDofFunc (
            HYPRE_Solver solver,
            HYPRE_Int * dof_func )
```

(Optional) Sets the mapping that assigns the function to each variable, if using the systems version. If no assignment is made and the number of functions is k $>$ 1, the mapping generated is (0,1,...,k-1,0,1,...,k-1,...).

2.4.3.90 HYPRE_BoomerAMGSetDomainType()

```
HYPRE_Int HYPRE_BoomerAMGSetDomainType (
            HYPRE_Solver solver,
            HYPRE_Int domain_type )
```

(Optional) Defines the type of domain used for the Schwarz method. The following options exist for domain_type:

\begin{tabular}{|c|l|} \hline 0 & each point is a domain \ 1 & each node is a domain (only of interest in "systems" AMG) \ 2 & each domain is generated by agglomeration (default) \ \hline \end{tabular}

2.4.3.91 HYPRE_BoomerAMGSetDropTol()

```
HYPRE_Int HYPRE_BoomerAMGSetDropTol (
            HYPRE_Solver solver,
            HYPRE_Real drop_tol )
```

(Optional) Defines drop tolerance for PILUT. For further explanation see description of PILUT.

2.4.3.92 HYPRE_BoomerAMGSetEuBJ()

```
HYPRE_Int HYPRE_BoomerAMGSetEuBJ (
            HYPRE_Solver solver,
            HYPRE_Int eu_bj )
```

(Optional) Defines use of block jacobi ILUT for Euclid. For further explanation see description of Euclid.

2.4.3.93 HYPRE_BoomerAMGSetEuclidFile()

```
HYPRE_Int HYPRE_BoomerAMGSetEuclidFile (
            HYPRE_Solver solver,
            char * euclidfile )
```

(Optional) Defines name of an input file for Euclid parameters. For further explanation see description of Euclid.

2.4.3.94 HYPRE_BoomerAMGSetEuLevel()

```
HYPRE_Int HYPRE_BoomerAMGSetEuLevel (
            HYPRE_Solver solver,
            HYPRE_Int eu_level )
```

(Optional) Defines number of levels for ILU(k) in Euclid. For further explanation see description of Euclid.

2.4.3.95 HYPRE_BoomerAMGSetEuSparseA()

```
HYPRE_Int HYPRE_BoomerAMGSetEuSparseA (
            HYPRE_Solver solver,
            HYPRE_Real eu_sparse_A )
```

(Optional) Defines filter for ILU(k) for Euclid. For further explanation see description of Euclid.

2.4.3.96 HYPRE_BoomerAMGSetFCycle()

```
HYPRE_Int HYPRE_BoomerAMGSetFCycle (
            HYPRE_Solver solver,
            HYPRE_Int fcycle )
```

(Optional) Specifies the use of Full multigrid cycle. The default is 0.

2.4.3.97 HYPRE_BoomerAMGSetFilter()

```
HYPRE_Int HYPRE_BoomerAMGSetFilter (
            HYPRE_Solver solver,
            HYPRE_Real filter )
```

(Optional) Defines filter for ParaSAILS. For further explanation see description of ParaSAILS.

2.4.3.98 HYPRE_BoomerAMGSetFilterThresholdR()

```
HYPRE_Int HYPRE_BoomerAMGSetFilterThresholdR (
            HYPRE_Solver solver,
            HYPRE_Real filter_threshold )
```

2.4.3.99 HYPRE_BoomerAMGSetGMRESSwitchR()

```
HYPRE_Int HYPRE_BoomerAMGSetGMRESSwitchR (
            HYPRE_Solver solver,
            HYPRE_Int gmres_switch )
```

(Optional) Set local problem size at which GMRES is used over a direct solve in approximating ideal restriction. The default is 0.

2.4.3.100 HYPRE_BoomerAMGSetGridRelaxPoints()

```
HYPRE_Int HYPRE_BoomerAMGSetGridRelaxPoints (
            HYPRE_Solver solver,
            HYPRE_Int ** grid_relax_points )
```

2.4.3.101 HYPRE_BoomerAMGSetGridRelaxType()

```
HYPRE_Int HYPRE_BoomerAMGSetGridRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int * grid_relax_type )
```

(Optional) Defines which smoother is used on the fine and coarse grid, the up and down cycle.

Note: This routine will be phased out!!!! Use HYPRE_BoomerAMGSetRelaxType or HYPRE_BoomerAMGSetCycle↩ RelaxType instead.

2.4.3.102 HYPRE_BoomerAMGSetGSMG()

```
HYPRE_Int HYPRE_BoomerAMGSetGSMG (
            HYPRE_Solver solver,
            HYPRE_Int gsmg )
```

(Optional) Specifies the use of GSMG - geometrically smooth coarsening and interpolation. Currently any nonzero value for gsmg will lead to the use of GSMG. The default is 0, i.e. (GSMG is not used)

2.4.3.103 HYPRE_BoomerAMGSetInterpType()

```
HYPRE_Int HYPRE_BoomerAMGSetInterpType (
            HYPRE_Solver solver,
            HYPRE_Int interp_type )
```

(Optional) Defines which parallel interpolation operator is used. There are the following options for interp_type:

\begin{tabular}{|c|l|} \hline 0 & classical modified interpolation \ 1 & LS interpolation (for use with GSMG) \ 2 & classical modified interpolation for hyperbolic PDEs \ 3 & direct interpolation (with separation of weights) \ 4 & multipass interpolation \ 5 & multipass interpolation (with separation of weights) \ 6 & extended+i interpolation \ 7 & extended+i (if no common C neighbor) interpolation \ 8 & standard interpolation \ 9 & standard interpolation (with separation of weights) \ 10 & classical block interpolation (for use with nodal systems version only) \ 11 & classical block interpolation (for use with nodal systems version only) \ & with diagonalized diagonal blocks \ 12 & FF interpolation \ 13 & FF1 interpolation \ 14 & extended interpolation \ \hline \end{tabular}

The default is ext+i interpolation (interp_type 6) trunctated to at most 4 \ elements per row. (see HYPRE_Boomer↩ AMGSetPMaxElmts).

2.4.3.104 HYPRE_BoomerAMGSetInterpVecAbsQTrunc()

```
HYPRE_Int HYPRE_BoomerAMGSetInterpVecAbsQTrunc (
            HYPRE_Solver solver,
            HYPRE_Real q_trunc )
```

(Optional) Defines a truncation factor for Q, the additional columns added to the original interpolation matrix P, to reduce complexity. The default is no truncation.

2.4.3.105 HYPRE_BoomerAMGSetInterpVecQMax()

```
HYPRE_Int HYPRE_BoomerAMGSetInterpVecQMax (
            HYPRE_Solver solver,
            HYPRE_Int q_max )
```

(Optional) Defines the maximal elements per row for Q, the additional columns added to the original interpolation matrix P, to reduce complexity. The default is no truncation.

2.4.3.106 HYPRE_BoomerAMGSetInterpVectors()

```
HYPRE_Int HYPRE_BoomerAMGSetInterpVectors (
            HYPRE_Solver solver,
            HYPRE_Int num_vectors,
            HYPRE_ParVector * interp_vectors )
```

(Optional) Allows the user to incorporate additional vectors into the interpolation for systems AMG, e.g. rigid body modes for linear elasticity problems. This can only be used in context with nodal coarsening and still requires the user to choose an interpolation.

2.4.3.107 HYPRE_BoomerAMGSetInterpVecVariant()

```
HYPRE_Int HYPRE_BoomerAMGSetInterpVecVariant (
            HYPRE_Solver solver,
            HYPRE_Int var )
```

(Optional) Defines the interpolation variant used for HYPRE_BoomerAMGSetInterpVectors: \begin{tabular}{|c|l|} \hline 1 & GM approach 1 \ 2 & GM approach 2 (to be preferred over 1) \ 3 & LN approach \ \hline \end{tabular}

2.4.3.108 HYPRE_BoomerAMGSetIsTriangular()

```
HYPRE_Int HYPRE_BoomerAMGSetIsTriangular (
            HYPRE_Solver solver,
            HYPRE_Int is_triangular )
```

(Optional) Assumes the matrix is triangular in some ordering to speed up the setup time of approximate ideal restriction.

The default is 0.

2.4.3.109 HYPRE_BoomerAMGSetISType()

```
HYPRE_Int HYPRE_BoomerAMGSetISType (
            HYPRE_Solver solver,
            HYPRE_Int IS_type )
```

2.4.3.110 HYPRE_BoomerAMGSetJacobiTruncThreshold()

```
HYPRE_Int HYPRE_BoomerAMGSetJacobiTruncThreshold (
            HYPRE_Solver solver,
            HYPRE_Real jacobi_trunc_threshold )
```

2.4.3.111 HYPRE_BoomerAMGSetKeepTranspose()

```
HYPRE_Int HYPRE_BoomerAMGSetKeepTranspose (
            HYPRE_Solver solver,
            HYPRE_Int keepTranspose )
```

(Optional) If set to 1, the local interpolation transposes will be saved to use more efficient matvecs instead of matvecTs

2.4.3.112 HYPRE_BoomerAMGSetLevel()

```
HYPRE_Int HYPRE_BoomerAMGSetLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

(Optional) Defines number of levels for ParaSAILS. For further explanation see description of ParaSAILS.

2.4.3.113 HYPRE_BoomerAMGSetLevelNonGalerkinTol()

```
HYPRE_Int HYPRE_BoomerAMGSetLevelNonGalerkinTol (
            HYPRE_Solver solver,
            HYPRE_Real nongalerkin_tol,
            HYPRE_Int level )
```

(Optional) Defines the level specific non-Galerkin drop-tolerances for sparsifying coarse grid operators and thus reducing communication. A drop-tolerance of 0.0 means to skip doing non-Galerkin on that level. The maximum drop tolerance for a level is 1.0, although much smaller values such as 0.03 or 0.01 are recommended.

Note that if the user wants to set a specific tolerance on all levels, HYPRE_BooemrAMGSetNonGalerkinTol should be used. Individual levels can then be changed using this routine.

In general, it is safer to drop more aggressively on coarser levels. For instance, one could use 0.0 on the finest level, 0.01 on the second level and then using 0.05 on all remaining levels. The best way to achieve this is to set 0.05 on all levels with HYPRE_BoomerAMGSetNonGalerkinTol and then change the tolerance on level 0 to 0.0 and the tolerance on level 1 to 0.01 with HYPRE_BoomerAMGSetLevelNonGalerkinTol. Like many AMG parameters, these drop tolerances can be tuned. It is also common to delay the start of the non-Galerkin process further to a later level than level 1.

Parameters

solver	[IN] solver or preconditioner object to be applied.
nongalerkin_tol	[IN] level specific drop tolerance
level	[IN] level on which drop tolerance is used

2.4.3.114 HYPRE_BoomerAMGSetLevelOuterWt()

```
HYPRE_Int HYPRE_BoomerAMGSetLevelOuterWt (
```

```
            HYPRE_Solver solver,
            HYPRE_Real omega,
            HYPRE_Int level )
```

(Optional) Defines the outer relaxation weight for hybrid SOR or SSOR on the user defined level. Note that the finest level is denoted 0, the next coarser level 1, etc. For nonpositive omega, the parameter is determined on the given level as described for HYPRE_BoomerAMGSetOuterWt. The default is 1.

2.4.3.115 HYPRE_BoomerAMGSetLevelRelaxWt()

```
HYPRE_Int HYPRE_BoomerAMGSetLevelRelaxWt (
            HYPRE_Solver solver,
            HYPRE_Real relax_weight,
            HYPRE_Int level )
```

(Optional) Defines the relaxation weight for smoothed Jacobi and hybrid SOR on the user defined level. Note that the finest level is denoted 0, the next coarser level 1, etc. For nonpositive relax_weight, the parameter is determined on the given level as described for HYPRE_BoomerAMGSetRelaxWt. The default is 1.

2.4.3.116 HYPRE_BoomerAMGSetLogging()

```
HYPRE_Int HYPRE_BoomerAMGSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Requests additional computations for diagnostic and similar data to be logged by the user. Default to 0 for do nothing. The latest residual will be available if logging $>$ 1.

2.4.3.117 HYPRE_BoomerAMGSetMaxCoarseSize()

```
HYPRE_Int HYPRE_BoomerAMGSetMaxCoarseSize (
            HYPRE_Solver solver,
            HYPRE_Int max_coarse_size )
```

(Optional) Sets maximum size of coarsest grid. The default is 9.

2.4.3.118 HYPRE_BoomerAMGSetMaxIter()

```
HYPRE_Int HYPRE_BoomerAMGSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Sets maximum number of iterations, if BoomerAMG is used as a solver. If it is used as a preconditioner, it should be set to 1. The default is 20.

2.4.3.119 HYPRE_BoomerAMGSetMaxLevels()

```
HYPRE_Int HYPRE_BoomerAMGSetMaxLevels (
            HYPRE_Solver solver,
            HYPRE_Int max_levels )
```

(Optional) Sets maximum number of multigrid levels. The default is 25.

2.4.3.120 HYPRE_BoomerAMGSetMaxNzPerRow()

```
HYPRE_Int HYPRE_BoomerAMGSetMaxNzPerRow (
            HYPRE_Solver solver,
            HYPRE_Int max_nz_per_row )
```

(Optional) Defines maximal number of nonzeros for PILUT. For further explanation see description of PILUT.

2.4.3.121 HYPRE_BoomerAMGSetMaxRowSum()

```
HYPRE_Int HYPRE_BoomerAMGSetMaxRowSum (
            HYPRE_Solver solver,
            HYPRE_Real max_row_sum )
```

(Optional) Sets a parameter to modify the definition of strength for diagonal dominant portions of the matrix. The default is 0.9. If max_row_sum is 1, no checking for diagonally dominant rows is performed.

2.4.3.122 HYPRE_BoomerAMGSetMeasureType()

```
HYPRE_Int HYPRE_BoomerAMGSetMeasureType (
            HYPRE_Solver solver,
            HYPRE_Int measure_type )
```

(Optional) Defines whether local or global measures are used.

2.4.3.123 HYPRE_BoomerAMGSetMinCoarseSize()

```
HYPRE_Int HYPRE_BoomerAMGSetMinCoarseSize (
            HYPRE_Solver solver,
            HYPRE_Int min_coarse_size )
```

(Optional) Sets minimum size of coarsest grid. The default is 1.

2.4.3.124 HYPRE_BoomerAMGSetMinIter()

```
HYPRE_Int HYPRE_BoomerAMGSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

(Optional)

2.4.3.125 HYPRE_BoomerAMGSetModuleRAP2()

```
HYPRE_Int HYPRE_BoomerAMGSetModuleRAP2 (
            HYPRE_Solver solver,
            HYPRE_Int mod_rap2 )
```

(Optional) If mod_rap2 not equal 0, the triple matrix product RAP is replaced by two matrix products with modularized kernels

2.4.3.126 HYPRE_BoomerAMGSetMultAdditive()

```
HYPRE_Int HYPRE_BoomerAMGSetMultAdditive (
            HYPRE_Solver solver,
            HYPRE_Int addlvl )
```

(Optional) Defines use of an additive V(1,1)-cycle using the mult-additive method starting at level 'addlvl'. The multiplicative approach is used on levels 0, ...'addlvl+1'. 'addlvl' needs to be $>$ -1 for this to have an effect. Can only be used with weighted Jacobi and l1-Jacobi(default).

Can only be used when AMG is used as a preconditioner !!!

2.4.3.127 HYPRE_BoomerAMGSetMultAddPMaxElmts()

```
HYPRE_Int HYPRE_BoomerAMGSetMultAddPMaxElmts (
            HYPRE_Solver solver,
            HYPRE_Int add_P_max_elmts )
```

(Optional) Defines the maximal number of elements per row for the smoothed interpolation used for mult-additive or simple method. The default is 0.

2.4.3.128 HYPRE_BoomerAMGSetMultAddTruncFactor()

```
HYPRE_Int HYPRE_BoomerAMGSetMultAddTruncFactor (
            HYPRE_Solver solver,
            HYPRE_Real add_trunc_factor )
```

(Optional) Defines the truncation factor for the smoothed interpolation used for mult-additive or simple method. The default is 0.

2.4.3.129 HYPRE_BoomerAMGSetNodal()

```
HYPRE_Int HYPRE_BoomerAMGSetNodal (
            HYPRE_Solver solver,
            HYPRE_Int nodal )
```

(Optional) Sets whether to use the nodal systems coarsening. Should be used for linear systems generated from systems of PDEs. The default is 0 (unknown-based coarsening, only coarsens within same function). For the remaining options a nodal matrix is generated by applying a norm to the nodal blocks and applying the coarsening algorithm to this matrix. \begin{tabular}{|c|l|} \hline 1 & Frobenius norm \ 2 & sum of absolute values of elements in each block \ 3 & largest element in each block (not absolute value)\ 4 & row-sum norm \ 6 & sum of all values in each block \ \hline \end{tabular}

2.4.3.130 HYPRE_BoomerAMGSetNodalDiag()

```
HYPRE_Int HYPRE_BoomerAMGSetNodalDiag (
            HYPRE_Solver solver,
            HYPRE_Int nodal_diag )
```

(Optional) Sets whether to give special treatment to diagonal elements in the nodal systems version. The default is 0. If set to 1, the diagonal entry is set to the negative sum of all off diagonal entries. If set to 2, the signs of all diagonal entries are inverted.

2.4.3.131 HYPRE_BoomerAMGSetNonGalerkinTol()

```
HYPRE_Int HYPRE_BoomerAMGSetNonGalerkinTol (
            HYPRE_Solver solver,
            HYPRE_Real nongalerkin_tol )
```

(Optional) Defines the non-Galerkin drop-tolerance for sparsifying coarse grid operators and thus reducing communication. Value specified here is set on all levels. This routine should be used before HYPRE_BoomerAMGSetLevelNon↵GalerkinTol, which then can be used to change individual levels if desired

2.4.3.132 HYPRE_BoomerAMGSetNonGalerkTol()

```
HYPRE_Int HYPRE_BoomerAMGSetNonGalerkTol (
            HYPRE_Solver solver,
            HYPRE_Int nongalerk_num_tol,
            HYPRE_Real * nongalerk_tol )
```

2.4.3.133 HYPRE_BoomerAMGSetNumCRRelaxSteps()

```
HYPRE_Int HYPRE_BoomerAMGSetNumCRRelaxSteps (
            HYPRE_Solver solver,
            HYPRE_Int num_CR_relax_steps )
```

2.4.3.134 HYPRE_BoomerAMGSetNumFunctions()

```
HYPRE_Int HYPRE_BoomerAMGSetNumFunctions (
            HYPRE_Solver solver,
            HYPRE_Int num_functions )
```

(Optional) Sets the size of the system of PDEs, if using the systems version. The default is 1, i.e. a scalar system.

2.4.3.135 HYPRE_BoomerAMGSetNumGridSweeps()

```
HYPRE_Int HYPRE_BoomerAMGSetNumGridSweeps (
            HYPRE_Solver solver,
            HYPRE_Int * num_grid_sweeps )
```

2.4.3.136 HYPRE_BoomerAMGSetNumPaths()

```
HYPRE_Int HYPRE_BoomerAMGSetNumPaths (
            HYPRE_Solver solver,
            HYPRE_Int num_paths )
```

(Optional) Defines the degree of aggressive coarsening. The default is 1. Larger numbers lead to less aggressive coarsening.

2.4.3.137 HYPRE_BoomerAMGSetNumSamples()

```
HYPRE_Int HYPRE_BoomerAMGSetNumSamples (
            HYPRE_Solver solver,
            HYPRE_Int num_samples )
```

(Optional) Defines the number of sample vectors used in GSMG or LS interpolation.

2.4.3.138 HYPRE_BoomerAMGSetNumSweeps()

```
HYPRE_Int HYPRE_BoomerAMGSetNumSweeps (
            HYPRE_Solver solver,
            HYPRE_Int num_sweeps )
```

(Optional) Sets the number of sweeps. On the finest level, the up and the down cycle the number of sweeps are set to num_sweeps and on the coarsest level to 1. The default is 1.

2.4.3.139 HYPRE_BoomerAMGSetOldDefault()

```
HYPRE_Int HYPRE_BoomerAMGSetOldDefault (
            HYPRE_Solver solver )
```

Recovers old default for coarsening and interpolation, i.e Falgout coarsening and untruncated modified classical interpolation. This option might be preferred for 2 dimensional problems.

2.4.3.140 HYPRE_BoomerAMGSetOmega()

```
HYPRE_Int HYPRE_BoomerAMGSetOmega (
            HYPRE_Solver solver,
            HYPRE_Real * omega )
```

(Optional) Defines the outer relaxation weight for hybrid SOR. Note: This routine will be phased out!!!! Use HYPRE_↩
BoomerAMGSetOuterWt or HYPRE_BoomerAMGSetLevelOuterWt instead.

2.4.3.141 HYPRE_BoomerAMGSetOuterWt()

```
HYPRE_Int HYPRE_BoomerAMGSetOuterWt (
            HYPRE_Solver solver,
            HYPRE_Real omega )
```

(Optional) Defines the outer relaxation weight for hybrid SOR and SSOR on all levels.

\begin{tabular}{|l|l|} \hline omega $>$ 0 & this assigns the same outer relaxation weight omega on each level\ omega = -k & an outer relaxation weight is determined with at most k CG steps on each level \ & (this only makes sense for symmetric positive definite problems and smoothers, e.g. SSOR) \ \hline \end{tabular}

The default is 1.

2.4.3.142 HYPRE_BoomerAMGSetOverlap()

```
HYPRE_Int HYPRE_BoomerAMGSetOverlap (
            HYPRE_Solver solver,
            HYPRE_Int overlap )
```

(Optional) Defines the overlap for the Schwarz method. The following options exist for overlap:

\begin{tabular}{|c|l|} \hline 0 & no overlap \ 1 & minimal overlap (default) \ 2 & overlap generated by including all neighbors of domain boundaries \ \hline \end{tabular}

2.4.3.143 HYPRE_BoomerAMGSetPlotFileName()

```
HYPRE_Int HYPRE_BoomerAMGSetPlotFileName (
            HYPRE_Solver solver,
            const char * plotfilename )
```

2.4.3.144 HYPRE_BoomerAMGSetPlotGrids()

```
HYPRE_Int HYPRE_BoomerAMGSetPlotGrids (
            HYPRE_Solver solver,
            HYPRE_Int plotgrids )
```

2.4.3.145 HYPRE_BoomerAMGSetPMaxElmts()

```
HYPRE_Int HYPRE_BoomerAMGSetPMaxElmts (
            HYPRE_Solver solver,
            HYPRE_Int P_max_elmts )
```

(Optional) Defines the maximal number of elements per row for the interpolation. The default is 4. To turn off truncation, it needs to be set to 0.

2.4.3.146 HYPRE_BoomerAMGSetPostInterpType()

```
HYPRE_Int HYPRE_BoomerAMGSetPostInterpType (
            HYPRE_Solver solver,
            HYPRE_Int post_interp_type )
```

2.4.3.147 HYPRE_BoomerAMGSetPrintFileName()

```
HYPRE_Int HYPRE_BoomerAMGSetPrintFileName (
            HYPRE_Solver solver,
            const char * print_file_name )
```

2.4.3.148 HYPRE_BoomerAMGSetPrintLevel()

```
HYPRE_Int HYPRE_BoomerAMGSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

(Optional) Requests automatic printing of setup and solve information.

\begin{tabular}{|c|l|} \hline 0 & no printout (default) \ 1 & print setup information \ 2 & print solve information \ 3 & print both setup and solve information \ \hline \end{tabular}

Note, that if one desires to print information and uses BoomerAMG as a preconditioner, suggested print$_$level is 1 to avoid excessive output, and use print$_$level of solver for solve phase information.

2.4.3.149 HYPRE_BoomerAMGSetRAP2()

```
HYPRE_Int HYPRE_BoomerAMGSetRAP2 (
            HYPRE_Solver solver,
            HYPRE_Int rap2 )
```

(Optional) If rap2 not equal 0, the triple matrix product RAP is replaced by two matrix products.

2.4.3.150 HYPRE_BoomerAMGSetRedundant()

```
HYPRE_Int HYPRE_BoomerAMGSetRedundant (
            HYPRE_Solver solver,
            HYPRE_Int redundant )
```

(Optional) operates switch for redundancy. Needs to be used with HYPRE_BoomerAMGSetSeqThreshold. Default is 0, i.e. no redundancy.

2.4.3.151 HYPRE_BoomerAMGSetRelaxOrder()

```
HYPRE_Int HYPRE_BoomerAMGSetRelaxOrder (
            HYPRE_Solver solver,
            HYPRE_Int relax_order )
```

(Optional) Defines in which order the points are relaxed. There are the following options for relax_order:

\begin{tabular}{|c|l|} \hline 0 & the points are relaxed in natural or lexicographic order on each processor \ 1 & CF-relaxation is used, i.e on the fine grid and the down cycle the coarse points are relaxed first, \ & followed by the fine points; on the up cycle the F-points are relaxed first, followed by the C-points. \ & On the coarsest level, if an iterative scheme is used, the points are relaxed in lexicographic order. \ \hline \end{tabular}

The default is 0.

2.4.3.152 HYPRE_BoomerAMGSetRelaxType()

```
HYPRE_Int HYPRE_BoomerAMGSetRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int relax_type )
```

(Optional) Defines the smoother to be used. It uses the given smoother on the fine grid, the up and the down cycle and sets the solver on the coarsest level to Gaussian elimination (9). The default is ℓ_1-Gauss-Seidel, forward solve (13) on the down cycle and backward solve (14) on the up cycle.

There are the following options for relax_type:

\begin{tabular}{|c|l|} \hline 0 & Jacobi \ 1 & Gauss-Seidel, sequential (very slow!) \ 2 & Gauss-Seidel, interior points in parallel, boundary sequential (slow!) \ 3 & hybrid Gauss-Seidel or SOR, forward solve \ 4 & hybrid Gauss-Seidel or SOR, backward solve \ 5 & hybrid chaotic Gauss-Seidel (works only with OpenMP) \ 6 & hybrid symmetric Gauss-Seidel or SSOR \ 8 & ℓ_1-scaled hybrid symmetric Gauss-Seidel\ 9 & Gaussian elimination (only on coarsest level) \ 13 & ℓ_1 Gauss-Seidel, forward solve \ 14 & ℓ_1 Gauss-Seidel, backward solve \ 15 & CG (warning - not a fixed smoother - may require FGMRES)\ 16 & Chebyshev\ 17 & FCF-Jacobi\ 18 & ℓ_1-scaled jacobi\ \hline \end{tabular}

2.4.3.153 HYPRE_BoomerAMGSetRelaxWeight()

```
HYPRE_Int HYPRE_BoomerAMGSetRelaxWeight (
            HYPRE_Solver solver,
            HYPRE_Real * relax_weight )
```

2.4.3.154 HYPRE_BoomerAMGSetRelaxWt()

```
HYPRE_Int HYPRE_BoomerAMGSetRelaxWt (
            HYPRE_Solver solver,
            HYPRE_Real relax_weight )
```

(Optional) Defines the relaxation weight for smoothed Jacobi and hybrid SOR on all levels.

\begin{tabular}{|l|l|} \hline relax_weight $>$ 0 & this assigns the given relaxation weight on all levels \ relax_weight = 0 & the weight is determined on each level with the estimate $3 \over {4|D^{-1/2}AD^{-1/2}|}$,\ & where D is the diagonal matrix of A (this should only be used with Jacobi) \ relax_weight = -k & the relaxation weight is determined with at most k CG steps on each level \ & this should only be used for symmetric positive definite problems) \ \hline \end{tabular}

The default is 1.

2.4.3.155 HYPRE_BoomerAMGSetRestriction()

```
HYPRE_Int HYPRE_BoomerAMGSetRestriction (
            HYPRE_Solver solver,
            HYPRE_Int restr_par )
```

(Optional) Defines which parallel restriction operator is used. There are the following options for restr_type:

\begin{tabular}{|c|l|} \hline 0 & P^T - Transpose of the interpolation operator \ 1 & AIR-1 - Approximate Ideal Restriction (distance 1) \ 2 & AIR-2 - Approximate Ideal Restriction (distance 2) \ \hline \end{tabular}

The default is 0.

2.4.3.156 HYPRE_BoomerAMGSetSabs()

```
HYPRE_Int HYPRE_BoomerAMGSetSabs (
            HYPRE_Solver solver,
            HYPRE_Int Sabs )
```

2.4.3.157 HYPRE_BoomerAMGSetSchwarzRlxWeight()

```
HYPRE_Int HYPRE_BoomerAMGSetSchwarzRlxWeight (
            HYPRE_Solver solver,
            HYPRE_Real schwarz_rlx_weight )
```

(Optional) Defines a smoothing parameter for the additive Schwarz method.

2.4.3.158 HYPRE_BoomerAMGSetSchwarzUseNonSymm()

```
HYPRE_Int HYPRE_BoomerAMGSetSchwarzUseNonSymm (
            HYPRE_Solver solver,
            HYPRE_Int use_nonsymm )
```

(Optional) Indicates that the aggregates may not be SPD for the Schwarz method. The following options exist for use_nonsymm:

\begin{tabular}{|c|l|} \hline 0 & assume SPD (default) \ 1 & assume non-symmetric \ \hline \end{tabular}

2.4.3.159 HYPRE_BoomerAMGSetSCommPkgSwitch()

```
HYPRE_Int HYPRE_BoomerAMGSetSCommPkgSwitch (
            HYPRE_Solver solver,
            HYPRE_Real S_commpkg_switch )
```

(Optional) Defines the largest strength threshold for which the strength matrix S uses the communication package of the operator A. If the strength threshold is larger than this values, a communication package is generated for S. This can save memory and decrease the amount of data that needs to be communicated, if S is substantially sparser than A. The default is 1.0.

2.4.3.160 HYPRE_BoomerAMGSetSepWeight()

```
HYPRE_Int HYPRE_BoomerAMGSetSepWeight (
            HYPRE_Solver solver,
            HYPRE_Int sep_weight )
```

(Optional) Defines whether separation of weights is used when defining strength for standard interpolation or multipass interpolation. Default: 0, i.e. no separation of weights used.

2.4.3.161 HYPRE_BoomerAMGSetSeqThreshold()

```
HYPRE_Int HYPRE_BoomerAMGSetSeqThreshold (
            HYPRE_Solver solver,
            HYPRE_Int seq_threshold )
```

(Optional) Sets maximal size for agglomeration or redundant coarse grid solve. When the system is smaller than this threshold, sequential AMG is used on process 0 or on all remaining active processes (if redundant = 1).

2.4.3.162 HYPRE_BoomerAMGSetSimple()

```
HYPRE_Int HYPRE_BoomerAMGSetSimple (
            HYPRE_Solver solver,
            HYPRE_Int addlvl )
```

(Optional) Defines use of an additive V(1,1)-cycle using the simplified mult-additive method starting at level 'addlvl'. The multiplicative approach is used on levels 0, ...'addlvl+1'. 'addlvl' needs to be > -1 for this to have an effect. Can only be used with weighted Jacobi and l1-Jacobi(default).

Can only be used when AMG is used as a preconditioner !!!

2.4.3.163 HYPRE_BoomerAMGSetSmoothNumLevels()

```
HYPRE_Int HYPRE_BoomerAMGSetSmoothNumLevels (
            HYPRE_Solver solver,
            HYPRE_Int smooth_num_levels )
```

(Optional) Sets the number of levels for more complex smoothers. The smoothers, as defined by HYPRE_BoomerAMGSetSmoothType, will be used on level 0 (the finest level) through level smooth_num_levels-1. The default is 0, i.e. no complex smoothers are used.

2.4.3.164 HYPRE_BoomerAMGSetSmoothNumSweeps()

```
HYPRE_Int HYPRE_BoomerAMGSetSmoothNumSweeps (
            HYPRE_Solver solver,
            HYPRE_Int smooth_num_sweeps )
```

(Optional) Sets the number of sweeps for more complex smoothers. The default is 1.

2.4.3.165 HYPRE_BoomerAMGSetSmoothType()

```
HYPRE_Int HYPRE_BoomerAMGSetSmoothType (
            HYPRE_Solver solver,
            HYPRE_Int smooth_type )
```

(Optional) Enables the use of more complex smoothers. The following options exist for smooth_type:

\begin{tabular}{|c|l|l|} \hline value & smoother & routines needed to set smoother parameters \ 6 & Schwarz smoothers & HYPRE_BoomerAMGSetDomainType, HYPRE_BoomerAMGSetOverlap, \ & & HYPRE_BoomerAMGSetVariant, HYPRE_BoomerAMGSetSchwarzRlxWeight \ 7 & Pilut & HYPRE_BoomerAMGSetDropTol, HYPRE_BoomerAMGSetMaxNzPerRow \ 8 & ParaSails & HYPRE_BoomerAMGSetSym, HYPRE_BoomerAMGSetLevel, \ & & HYPRE_BoomerAMGSetFilter, HYPRE_BoomerAMGSetThreshold \ 9 & Euclid & HYPRE_BoomerAMGSetEuclidFile \ \hline \end{tabular}

The default is 6. Also, if no smoother parameters are set via the routines mentioned in the table above, default values are used.

2.4.3.166 HYPRE_BoomerAMGSetStrongThreshold()

```
HYPRE_Int HYPRE_BoomerAMGSetStrongThreshold (
            HYPRE_Solver solver,
            HYPRE_Real strong_threshold )
```

(Optional) Sets AMG strength threshold. The default is 0.25. For 2d Laplace operators, 0.25 is a good value, for 3d Laplace operators, 0.5 or 0.6 is a better value. For elasticity problems, a large strength threshold, such as 0.9, is often better. The strong threshold for R is strong connections used in building an approximate ideal restriction, and the filter threshold for R a threshold to eliminate small entries from R after building it.

2.4.3.167 HYPRE_BoomerAMGSetStrongThresholdR()

```
HYPRE_Int HYPRE_BoomerAMGSetStrongThresholdR (
            HYPRE_Solver solver,
            HYPRE_Real strong_threshold )
```

2.4.3.168 HYPRE_BoomerAMGSetSym()

```
HYPRE_Int HYPRE_BoomerAMGSetSym (
            HYPRE_Solver solver,
            HYPRE_Int sym )
```

(Optional) Defines symmetry for ParaSAILS. For further explanation see description of ParaSAILS.

2.4.3.169 HYPRE_BoomerAMGSetThreshold()

```
HYPRE_Int HYPRE_BoomerAMGSetThreshold (
            HYPRE_Solver solver,
            HYPRE_Real threshold )
```

(Optional) Defines threshold for ParaSAILS. For further explanation see description of ParaSAILS.

2.4.3.170 HYPRE_BoomerAMGSetTol()

```
HYPRE_Int HYPRE_BoomerAMGSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance, if BoomerAMG is used as a solver. If it is used as a preconditioner, it should be set to 0. The default is 1.e-7.

2.4.3.171 HYPRE_BoomerAMGSetTruncFactor()

```
HYPRE_Int HYPRE_BoomerAMGSetTruncFactor (
            HYPRE_Solver solver,
            HYPRE_Real trunc_factor )
```

(Optional) Defines a truncation factor for the interpolation. The default is 0.

2.4.3.172 HYPRE_BoomerAMGSetup()

```
HYPRE_Int HYPRE_BoomerAMGSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Set up the BoomerAMG solver or preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] object to be set up.
A	[IN] ParCSR matrix used to construct the solver/preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.173 HYPRE_BoomerAMGSetVariant()

```
HYPRE_Int HYPRE_BoomerAMGSetVariant (
            HYPRE_Solver solver,
            HYPRE_Int variant )
```

(Optional) Defines which variant of the Schwarz method is used. The following options exist for variant:

\begin{tabular}{|c|l|} \hline 0 & hybrid multiplicative Schwarz method (no overlap across processor boundaries) \ 1 & hybrid additive Schwarz method (no overlap across processor boundaries) \ 2 & additive Schwarz method \ 3 & hybrid multiplicative Schwarz method (with overlap across processor boundaries) \ \hline \end{tabular}

The default is 0.

2.4.3.174 HYPRE_BoomerAMGSolve()

```
HYPRE_Int HYPRE_BoomerAMGSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve the system or apply AMG as a preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix, matrix of the linear system to be solved
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.175 HYPRE_BoomerAMGSolveT()

```
HYPRE_Int HYPRE_BoomerAMGSolveT (
```

```
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve the transpose system $A^T x = b$ or apply AMG as a preconditioner to the transpose system . Note that this function should only be used when preconditioning CGNR with BoomerAMG. It can only be used with Jacobi smoothing (relax_type 0 or 7) and without CF smoothing, i.e relax_order needs to be set to 0. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.176 HYPRE_EuclidCreate()

```
HYPRE_Int HYPRE_EuclidCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a Euclid object.

2.4.3.177 HYPRE_EuclidDestroy()

```
HYPRE_Int HYPRE_EuclidDestroy (
            HYPRE_Solver solver )
```

Destroy a Euclid object.

2.4.3.178 HYPRE_EuclidSetBJ()

```
HYPRE_Int HYPRE_EuclidSetBJ (
            HYPRE_Solver solver,
            HYPRE_Int bj )
```

Use block Jacobi ILU preconditioning instead of PILU

2.4.3.179 HYPRE_EuclidSetILUT()

```
HYPRE_Int HYPRE_EuclidSetILUT (
            HYPRE_Solver solver,
            HYPRE_Real drop_tol )
```

uses ILUT and defines a drop tolerance relative to the largest absolute value of any entry in the row being factored.

2.4.3.180 HYPRE_EuclidSetLevel()

```
HYPRE_Int HYPRE_EuclidSetLevel (
            HYPRE_Solver solver,
            HYPRE_Int level )
```

Set level k for ILU(k) factorization, default: 1

2.4.3.181 HYPRE_EuclidSetMem()

```
HYPRE_Int HYPRE_EuclidSetMem (
            HYPRE_Solver solver,
            HYPRE_Int eu_mem )
```

If eu_mem not equal 0, a summary of Euclid's memory usage is printed to stdout.

2.4.3.182 HYPRE_EuclidSetParams()

```
HYPRE_Int HYPRE_EuclidSetParams (
            HYPRE_Solver solver,
            HYPRE_Int argc,
            char * argv[] )
```

Insert (name, value) pairs in Euclid's options database by passing Euclid the command line (or an array of strings). All Euclid options (e.g, level, drop-tolerance) are stored in this database. If a (name, value) pair already exists, this call updates the value. See also: HYPRE_EuclidSetParamsFromFile.

Parameters

argc	[IN] Length of argv array
argv	[IN] Array of strings

2.4.3.183 HYPRE_EuclidSetParamsFromFile()

```
HYPRE_Int HYPRE_EuclidSetParamsFromFile (
            HYPRE_Solver solver,
            char * filename )
```

Insert (name, value) pairs in Euclid's options database. Each line of the file should either begin with a `‘#,'' indicating a comment line, or contain a (name value) pair, e.g: \

>cat optionsFile \ #sample runtime parameter file \ -blockJacobi 3 \ -matFile /home/hysom/myfile.euclid \ -doSomething true \ -xx_coeff -1.0

See also: HYPRE_EuclidSetParams.

Parameters

filename[IN]	Pathname/filename to read

2.4.3.184 HYPRE_EuclidSetRowScale()

```
HYPRE_Int HYPRE_EuclidSetRowScale (
            HYPRE_Solver solver,
            HYPRE_Int row_scale )
```

If row_scale not equal 0, values are scaled prior to factorization so that largest value in any row is +1 or -1. Note that this can destroy symmetry in a matrix.

2.4.3.185 HYPRE_EuclidSetSparseA()

```
HYPRE_Int HYPRE_EuclidSetSparseA (
            HYPRE_Solver solver,
            HYPRE_Real sparse_A )
```

Defines a drop tolerance for ILU(k). Default: 0 Use with HYPRE_EuclidSetRowScale. Note that this can destroy symmetry in a matrix.

2.4.3.186 HYPRE_EuclidSetStats()

```
HYPRE_Int HYPRE_EuclidSetStats (
            HYPRE_Solver solver,
            HYPRE_Int eu_stats )
```

If eu_stats not equal 0, a summary of runtime settings and timing information is printed to stdout.

2.4.3.187 HYPRE_EuclidSetup()

```
HYPRE_Int HYPRE_EuclidSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Set up the Euclid preconditioner. This function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] Preconditioner object to set up.
A	[IN] ParCSR matrix used to construct the preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.188 HYPRE_EuclidSolve()

```
HYPRE_Int HYPRE_EuclidSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Apply the Euclid preconditioner. This function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] Preconditioner object to apply.
A	Ignored by this function.
b	[IN] Vector to precondition.
x	[OUT] Preconditioned vector.

2.4.3.189 HYPRE_MGRCreate()

```
HYPRE_Int HYPRE_MGRCreate (
            HYPRE_Solver * solver )
```

Create a solver object

2.4.3.190 HYPRE_MGRDestroy()

```
HYPRE_Int HYPRE_MGRDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object

2.4.3.191 HYPRE_MGRGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_MGRGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * res_norm )
```

(Optional) Return the norm of the final relative residual.

2.4.3.192 HYPRE_MGRGetNumIterations()

```
HYPRE_Int HYPRE_MGRGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

(Optional) Return the number of MGR iterations.

2.4.3.193 HYPRE_MGRSetBlockSize()

```
HYPRE_Int HYPRE_MGRSetBlockSize (
            HYPRE_Solver solver,
            HYPRE_Int bsize )
```

(Optional) Set the system block size. This should match the block size set in the MGRSetCpointsByBlock function. The default is 1.

2.4.3.194 HYPRE_MGRSetCoarseSolver()

```
HYPRE_Int HYPRE_MGRSetCoarseSolver (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn coarse_grid_solver_solve,
            HYPRE_PtrToParSolverFcn coarse_grid_solver_setup,
            HYPRE_Solver coarse_grid_solver )
```

(Optional) Set the coarse grid solver. Currently uses BoomerAMG. The default, if not set, is BoomerAMG with default options.

Parameters

solver	[IN] solver or preconditioner object
coarse_grid_solver_solve	[IN] solve routine for BoomerAMG
coarse_grid_solver_setup	[IN] setup routine for BoomerAMG
coarse_grid_solver	[IN] BoomerAMG solver

2.4.3.195 HYPRE_MGRSetCpointsByBlock()

```
HYPRE_Int HYPRE_MGRSetCpointsByBlock (
            HYPRE_Solver solver,
            HYPRE_Int block_size,
            HYPRE_Int max_num_levels,
            HYPRE_Int * num_block_coarse_points,
            HYPRE_Int ** block_coarse_indexes )
```

Set the block data and prescribe the coarse indexes per block for each reduction level.

Parameters

solver	[IN] solver or preconditioner object
block_size	[IN] system block size
max_num_levels	[IN] maximum number of reduction levels
num_block_coarse_points	[IN] number of coarse points per block per level
block_coarse_indexes	[IN] index for each block coarse point per level

2.4.3.196 HYPRE_MGRSetFRelaxMethod()

```
HYPRE_Int HYPRE_MGRSetFRelaxMethod (
            HYPRE_Solver solver,
            HYPRE_Int relax_method )
```

(Optional) Set the strategy for F-relaxation. Options for {\tt relax_method} are:

\begin{tabular}{|c|l|} \hline 0 & Single-level relaxation sweeps for F-relaxation as prescribed by {\tt MGRSetRelaxType} \ 1 & Multi-level relaxation strategy for F-relaxation (V(1,0) cycle currently supported). \ \hline \end{tabular}

2.4.3.197 HYPRE_MGRSetGlobalsmoothType()

```
HYPRE_Int HYPRE_MGRSetGlobalsmoothType (
            HYPRE_Solver solver,
            HYPRE_Int smooth_type )
```

(Optional) Determines type of global smoother. Options for {\tt smooth_type} are:

\begin{tabular}{|c|l|} \hline 0 & block Jacobi (default) \ 1 & Jacobi \ 2 & Gauss-Seidel, sequential (very slow!) \ 3 & Gauss-Seidel, interior points in parallel, boundary sequential (slow!) \ 4 & hybrid Gauss-Seidel or SOR, forward solve \ 5 & hybrid Gauss-Seidel or SOR, backward solve \ 6 & hybrid chaotic Gauss-Seidel (works only with OpenMP) \ 7 & hybrid symmetric Gauss-Seidel or SSOR \ 8 & Euclid (ILU) \ \hline \end{tabular}

2.4.3.198 HYPRE_MGRSetInterpType()

```
HYPRE_Int HYPRE_MGRSetInterpType (
            HYPRE_Solver solver,
            HYPRE_Int interp_type )
```

(Optional) Set the strategy for computing the MGR restriction operator. Options for {\tt interp_type} are:

\begin{tabular}{|c|l|} \hline 0 & injection $[0 I]^{T}$ \ 1 & unscaled (not recommended) \ 2 & diagonal scaling (Jacobi) \ else & use default (classical modified interpolation) \ \hline \end{tabular}

These options are currently active for the last stage reduction. Intermediate reduction levels use diagonal scaling.

2.4.3.199 HYPRE_MGRSetLogging()

```
HYPRE_Int HYPRE_MGRSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

(Optional) Requests logging of solver diagnostics. Requests additional computations for diagnostic and similar data to be logged by the user. Default to 0 for do nothing. The latest residual will be available if logging $>$ 1.

2.4.3.200 HYPRE_MGRSetMaxCoarseLevels()

```
HYPRE_Int HYPRE_MGRSetMaxCoarseLevels (
            HYPRE_Solver solver,
            HYPRE_Int maxlev )
```

(Optional) Set maximum number of coarsening (or reduction) levels. The default is 10.

2.4.3.201 HYPRE_MGRSetMaxGlobalsmoothIters()

```
HYPRE_Int HYPRE_MGRSetMaxGlobalsmoothIters (
            HYPRE_Solver solver,
            HYPRE_Int smooth_iter )
```

(Optional) Determines how many sweeps of global smoothing to do. Default is 0 (no global smoothing).

2.4.3.202 HYPRE_MGRSetMaxIter()

```
HYPRE_Int HYPRE_MGRSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations if used as a solver. Set this to 1 if MGR is used as a preconditioner. The default is 20.

2.4.3.203 HYPRE_MGRSetNonCpointsToFpoints()

```
HYPRE_Int HYPRE_MGRSetNonCpointsToFpoints (
            HYPRE_Solver solver,
            HYPRE_Int nonCptToFptFlag )
```

(Optional) Set non C-points to F-points. This routine determines how the coarse points are selected for the next level reduction. Options for {\tt nonCptToFptFlag} are:

\begin{tabular}{|c|l|} \hline 0 & Allow points not prescribed as C points to be potentially set as C points \ & using classical AMG coarsening strategies (currently uses CLJP-coarsening). \ 1 & Fix points not prescribed as C points to be F points for the next reduction \ \hline \end{tabular}

2.4.3.204 HYPRE_MGRSetNumInterpSweeps()

```
HYPRE_Int HYPRE_MGRSetNumInterpSweeps (
            HYPRE_Solver solver,
            HYPRE_Int nsweeps )
```

(Optional) Set number of interpolation sweeps. This option is for interp_type $>$ 2.

2.4.3.205 HYPRE_MGRSetNumRelaxSweeps()

```
HYPRE_Int HYPRE_MGRSetNumRelaxSweeps (
            HYPRE_Solver solver,
            HYPRE_Int nsweeps )
```

(Optional) Set number of relaxation sweeps. This option is for the ‘single level’ F-relaxation (relax_method = 0).

2.4.3.206 HYPRE_MGRSetNumRestrictSweeps()

```
HYPRE_Int HYPRE_MGRSetNumRestrictSweeps (
            HYPRE_Solver solver,
            HYPRE_Int nsweeps )
```

(Optional) Set number of restriction sweeps. This option is for restrict_type $>$ 2.

2.4.3.207 HYPRE_MGRSetPrintLevel()

```
HYPRE_Int HYPRE_MGRSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

(Optional) Set the print level to print setup and solve information.

\begin{tabular}{|c|l|} \hline 0 & no printout (default) \ 1 & print setup information \ 2 & print solve information \ 3 & print both setup and solve information \ \hline \end{tabular}

2.4.3.208 HYPRE_MGRSetRelaxType()

```
HYPRE_Int HYPRE_MGRSetRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int relax_type )
```

(Optional) Set the relaxation type for F-relaxation. Currently supports the following flavors of relaxation types as described in the {\tt BoomerAMGSetRelaxType}: relax_types 0 - 8, 13, 14, 18, 19, 98.

2.4.3.209 HYPRE_MGRSetReservedCoarseNodes()

```
HYPRE_Int HYPRE_MGRSetReservedCoarseNodes (
            HYPRE_Solver solver,
            HYPRE_Int reserved_coarse_size,
            HYPRE_Int * reserved_coarse_nodes )
```

(Optional) Defines indexes of coarse nodes to be kept to the coarsest level. These indexes are passed down through the MGR hierarchy to the coarsest grid of the coarse grid (BoomerAMG) solver.

Parameters

solver	[IN] solver or preconditioner object
reserved_coarse_size	[IN] number of reserved coarse points
reserved_coarse_nodes	[IN] (global) indexes of reserved coarse points

2.4.3.210 HYPRE_MGRSetRestrictType()

```
HYPRE_Int HYPRE_MGRSetRestrictType (
            HYPRE_Solver solver,
            HYPRE_Int restrict_type )
```

(Optional) Set the strategy for computing the MGR restriction operator.

Options for {\tt restrict_type} are:

\begin{tabular}{|c|l|} \hline 0 & injection $[0 I]$ \ 1 & unscaled (not recommended) \ 2 & diagonal scaling (Jacobi) \ else & use classical modified interpolation \ \hline \end{tabular}

These options are currently active for the last stage reduction. Intermediate reduction levels use injection. The default is injection.

2.4.3.211 HYPRE_MGRSetTol()

```
HYPRE_Int HYPRE_MGRSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance for the MGR solver. Use tol = 0.0 if MGR is used as a preconditioner. The default is 1.e-7.

2.4.3.212 HYPRE_MGRSetup()

```
HYPRE_Int HYPRE_MGRSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Setup the MGR solver or preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] object to be set up.
A	[IN] ParCSR matrix used to construct the solver/preconditioner.
b	right-hand-side of the linear system to be solved (Ignored by this function).
x	approximate solution of the linear system to be solved (Ignored by this function).

2.4.3.213 HYPRE_MGRSolve()

```
HYPRE_Int HYPRE_MGRSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve the system or apply MGR as a preconditioner. If used as a preconditioner, this function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix, matrix of the linear system to be solved
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.214 HYPRE_ParaSailsBuildIJMatrix()

```
HYPRE_Int HYPRE_ParaSailsBuildIJMatrix (
            HYPRE_Solver solver,
            HYPRE_IJMatrix * pij_A )
```

Build IJ Matrix of the sparse approximate inverse (factor). This function explicitly creates the IJ Matrix corresponding to the sparse approximate inverse or the inverse factor. Example: HYPRE_IJMatrix ij_A; HYPRE_ParaSailsBuild↩ IJMatrix(solver, &ij_A);

Parameters

solver	[IN] Preconditioner object.
pij_A	[OUT] Pointer to the IJ Matrix.

2.4.3.215 HYPRE_ParaSailsCreate()

```
HYPRE_Int HYPRE_ParaSailsCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a ParaSails preconditioner.

2.4.3.216 HYPRE_ParaSailsDestroy()

```
HYPRE_Int HYPRE_ParaSailsDestroy (
            HYPRE_Solver solver )
```

Destroy a ParaSails preconditioner.

2.4.3.217 HYPRE_ParaSailsSetFilter()

```
HYPRE_Int HYPRE_ParaSailsSetFilter (
            HYPRE_Solver solver,
            HYPRE_Real filter )
```

Set the filter parameter for the ParaSails preconditioner.

Parameters

solver	[IN] Preconditioner object for which to set filter parameter.
filter	[IN] Value of filter parameter. The filter parameter is used to drop small nonzeros in the preconditioner, to reduce the cost of applying the preconditioner. Values from 0.05 to 0.1 are recommended. The default value is 0.1.

2.4.3.218 HYPRE_ParaSailsSetLoadbal()

```
HYPRE_Int HYPRE_ParaSailsSetLoadbal (
            HYPRE_Solver solver,
            HYPRE_Real loadbal )
```

Set the load balance parameter for the ParaSails preconditioner.

Parameters

solver	[IN] Preconditioner object for which to set the load balance parameter.
loadbal	[IN] Value of the load balance parameter, $0 \le$ loadbal ≤ 1. A zero value indicates that no load balance is attempted; a value of unity indicates that perfect load balance will be attempted. The recommended value is 0.9 to balance the overhead of data exchanges for load balancing. No load balancing is needed if the preconditioner is very sparse and fast to construct. The default value when this parameter is not set is 0.

2.4.3.219 HYPRE_ParaSailsSetLogging()

```
HYPRE_Int HYPRE_ParaSailsSetLogging (
```

```
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

Set the logging parameter for the ParaSails preconditioner.

Parameters

solver	[IN] Preconditioner object for which to set the logging parameter.
logging	[IN] Value of the logging parameter. A nonzero value sends statistics of the setup procedure to stdout. The default value when this parameter is not set is 0.

2.4.3.220 HYPRE_ParaSailsSetParams()

```
HYPRE_Int HYPRE_ParaSailsSetParams (
            HYPRE_Solver solver,
            HYPRE_Real thresh,
            HYPRE_Int nlevels )
```

Set the threshold and levels parameter for the ParaSails preconditioner. The accuracy and cost of ParaSails are parameterized by these two parameters. Lower values of the threshold parameter and higher values of levels parameter lead to more accurate, but more expensive preconditioners.

Parameters

solver	[IN] Preconditioner object for which to set parameters.
thresh	[IN] Value of threshold parameter, $0 \le$ thresh ≤ 1. The default value is 0.1.
nlevels	[IN] Value of levels parameter, $0 \le$ nlevels. The default value is 1.

2.4.3.221 HYPRE_ParaSailsSetReuse()

```
HYPRE_Int HYPRE_ParaSailsSetReuse (
            HYPRE_Solver solver,
            HYPRE_Int reuse )
```

Set the pattern reuse parameter for the ParaSails preconditioner.

Parameters

solver	[IN] Preconditioner object for which to set the pattern reuse parameter.
reuse	[IN] Value of the pattern reuse parameter. A nonzero value indicates that the pattern of the preconditioner should be reused for subsequent constructions of the preconditioner. A zero value indicates that the preconditioner should be constructed from scratch. The default value when this parameter is not set is 0.

2.4.3.222 HYPRE_ParaSailsSetSym()

```
HYPRE_Int HYPRE_ParaSailsSetSym (
            HYPRE_Solver solver,
            HYPRE_Int sym )
```

Set the symmetry parameter for the ParaSails preconditioner.

\begin{tabular}{|c|l|} \hline value & meaning \ \hline 0 & nonsymmetric and/or indefinite problem, and nonsymmetric preconditioner\ 1 & SPD problem, and SPD (factored) preconditioner \ 2 & nonsymmetric, definite problem, and SPD (factored) preconditioner \ \hline \end{tabular}

Parameters

solver	[IN] Preconditioner object for which to set symmetry parameter.
sym	[IN] Symmetry parameter.

2.4.3.223 HYPRE_ParaSailsSetup()

```
HYPRE_Int HYPRE_ParaSailsSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Set up the ParaSails preconditioner. This function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] Preconditioner object to set up.
A	[IN] ParCSR matrix used to construct the preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.224 HYPRE_ParaSailsSolve()

```
HYPRE_Int HYPRE_ParaSailsSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
```

```
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Apply the ParaSails preconditioner. This function should be passed to the iterative solver {\tt SetPrecond} function.

Parameters

solver	[IN] Preconditioner object to apply.
A	Ignored by this function.
b	[IN] Vector to precondition.
x	[OUT] Preconditioned vector.

2.4.3.225 HYPRE_ParCSRBiCGSTABCreate()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object

2.4.3.226 HYPRE_ParCSRBiCGSTABDestroy()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.227 HYPRE_ParCSRBiCGSTABGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.228 HYPRE_ParCSRBiCGSTABGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.229 HYPRE_ParCSRBiCGSTABGetPrecond()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.230 HYPRE_ParCSRBiCGSTABGetResidual()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

2.4.3.231 HYPRE_ParCSRBiCGSTABSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

2.4.3.232 HYPRE_ParCSRBiCGSTABSetLogging()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.233 HYPRE_ParCSRBiCGSTABSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.234 HYPRE_ParCSRBiCGSTABSetMinIter()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.235 HYPRE_ParCSRBiCGSTABSetPrecond()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.236 HYPRE_ParCSRBiCGSTABSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.237 HYPRE_ParCSRBiCGSTABSetStopCrit()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.4.3.238 HYPRE_ParCSRBiCGSTABSetTol()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.239 HYPRE_ParCSRBiCGSTABSetup()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.240 HYPRE_ParCSRBiCGSTABSolve()

```
HYPRE_Int HYPRE_ParCSRBiCGSTABSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.241 HYPRE_ParCSRCGNRCreate()

```
HYPRE_Int HYPRE_ParCSRCGNRCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

2.4.3.242 HYPRE_ParCSRCGNRDestroy()

```
HYPRE_Int HYPRE_ParCSRCGNRDestroy (
            HYPRE_Solver solver )
```

2.4.3.243 HYPRE_ParCSRCGNRGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRCGNRGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.244 HYPRE_ParCSRCGNRGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRCGNRGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.245 HYPRE_ParCSRCGNRGetPrecond()

```
HYPRE_Int HYPRE_ParCSRCGNRGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.246 HYPRE_ParCSRCGNRSetLogging()

```
HYPRE_Int HYPRE_ParCSRCGNRSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.247 HYPRE_ParCSRCGNRSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRCGNRSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.248 HYPRE_ParCSRCGNRSetMinIter()

```
HYPRE_Int HYPRE_ParCSRCGNRSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.249 HYPRE_ParCSRCGNRSetPrecond()

```
HYPRE_Int HYPRE_ParCSRCGNRSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precondT,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.250 HYPRE_ParCSRCGNRSetStopCrit()

```
HYPRE_Int HYPRE_ParCSRCGNRSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.4.3.251 HYPRE_ParCSRCGNRSetTol()

```
HYPRE_Int HYPRE_ParCSRCGNRSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.252 HYPRE_ParCSRCGNRSetup()

```
HYPRE_Int HYPRE_ParCSRCGNRSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.253 HYPRE_ParCSRCGNRSolve()

```
HYPRE_Int HYPRE_ParCSRCGNRSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.254 HYPRE_ParCSRCOGMRESCreate()

```
HYPRE_Int HYPRE_ParCSRCOGMRESCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.255 HYPRE_ParCSRCOGMRESDestroy()

```
HYPRE_Int HYPRE_ParCSRCOGMRESDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.256 HYPRE_ParCSRCOGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRCOGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.257 HYPRE_ParCSRCOGMRESGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRCOGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.258 HYPRE_ParCSRCOGMRESGetPrecond()

```
HYPRE_Int HYPRE_ParCSRCOGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.259 HYPRE_ParCSRCOGMRESGetResidual()

```
HYPRE_Int HYPRE_ParCSRCOGMRESGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

Returns the residual.

2.4.3.260 HYPRE_ParCSRCOGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

2.4.3.261 HYPRE_ParCSRCOGMRESSetCGS()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetCGS (
            HYPRE_Solver solver,
            HYPRE_Int cgs )
```

2.4.3.262 HYPRE_ParCSRCOGMRESSetKDim()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

2.4.3.263 HYPRE_ParCSRCOGMRESSetLogging()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.264 HYPRE_ParCSRCOGMRESSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.265 HYPRE_ParCSRCOGMRESSetMinIter()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.266 HYPRE_ParCSRCOGMRESSetPrecond()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.267 HYPRE_ParCSRCOGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.268 HYPRE_ParCSRCOGMRESSetTol()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.269 HYPRE_ParCSRCOGMRESSetUnroll()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetUnroll (
            HYPRE_Solver solver,
            HYPRE_Int unroll )
```

2.4.3.270 HYPRE_ParCSRCOGMRESSetup()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.271 HYPRE_ParCSRCOGMRESSolve()

```
HYPRE_Int HYPRE_ParCSRCOGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.272 HYPRE_ParCSRDiagScale()

```
HYPRE_Int HYPRE_ParCSRDiagScale (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix HA,
            HYPRE_ParVector Hy,
            HYPRE_ParVector Hx )
```

Solve routine for diagonal preconditioning.

2.4.3.273 HYPRE_ParCSRDiagScaleSetup()

```
HYPRE_Int HYPRE_ParCSRDiagScaleSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector y,
            HYPRE_ParVector x )
```

Setup routine for diagonal preconditioning.

2.4.3.274 HYPRE_ParCSRFlexGMRESCreate()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.275 HYPRE_ParCSRFlexGMRESDestroy()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.276 HYPRE_ParCSRFlexGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.277 HYPRE_ParCSRFlexGMRESGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.278 HYPRE_ParCSRFlexGMRESGetPrecond()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.279 HYPRE_ParCSRFlexGMRESGetResidual()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

2.4.3.280 HYPRE_ParCSRFlexGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

2.4.3.281 HYPRE_ParCSRFlexGMRESSetKDim()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

2.4.3.282 HYPRE_ParCSRFlexGMRESSetLogging()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.283 HYPRE_ParCSRFlexGMRESSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.284 HYPRE_ParCSRFlexGMRESSetMinIter()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.285 HYPRE_ParCSRFlexGMRESSetModifyPC()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetModifyPC (
            HYPRE_Solver solver,
            HYPRE_PtrToModifyPCFcn modify_pc )
```

2.4.3.286 HYPRE_ParCSRFlexGMRESSetPrecond()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.287 HYPRE_ParCSRFlexGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.288 HYPRE_ParCSRFlexGMRESSetTol()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.289 HYPRE_ParCSRFlexGMRESSetup()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.290 HYPRE_ParCSRFlexGMRESSolve()

```
HYPRE_Int HYPRE_ParCSRFlexGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.291 HYPRE_ParCSRGMRESCreate()

```
HYPRE_Int HYPRE_ParCSRGMRESCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.292 HYPRE_ParCSRGMRESDestroy()

```
HYPRE_Int HYPRE_ParCSRGMRESDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.293 HYPRE_ParCSRGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.294 HYPRE_ParCSRGMRESGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.295 HYPRE_ParCSRGMRESGetPrecond()

```
HYPRE_Int HYPRE_ParCSRGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.296 HYPRE_ParCSRGMRESGetResidual()

```
HYPRE_Int HYPRE_ParCSRGMRESGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

Returns the residual.

2.4.3.297 HYPRE_ParCSRGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

2.4.3.298 HYPRE_ParCSRGMRESSetKDim()

```
HYPRE_Int HYPRE_ParCSRGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

2.4.3.299 HYPRE_ParCSRGMRESSetLogging()

```
HYPRE_Int HYPRE_ParCSRGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.300 HYPRE_ParCSRGMRESSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.301 HYPRE_ParCSRGMRESSetMinIter()

```
HYPRE_Int HYPRE_ParCSRGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.302 HYPRE_ParCSRGMRESSetPrecond()

```
HYPRE_Int HYPRE_ParCSRGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.303 HYPRE_ParCSRGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.304 HYPRE_ParCSRGMRESSetStopCrit()

```
HYPRE_Int HYPRE_ParCSRGMRESSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.4.3.305 HYPRE_ParCSRGMRESSetTol()

```
HYPRE_Int HYPRE_ParCSRGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.306 HYPRE_ParCSRGMRESSetup()

```
HYPRE_Int HYPRE_ParCSRGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.307 HYPRE_ParCSRGMRESSolve()

```
HYPRE_Int HYPRE_ParCSRGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.308 HYPRE_ParCSRHybridCreate()

```
HYPRE_Int HYPRE_ParCSRHybridCreate (
            HYPRE_Solver * solver )
```

Create solver object

2.4.3.309 HYPRE_ParCSRHybridDestroy()

```
HYPRE_Int HYPRE_ParCSRHybridDestroy (
            HYPRE_Solver solver )
```

Destroy solver object

2.4.3.310 HYPRE_ParCSRHybridGetDSCGNumIterations()

```
HYPRE_Int HYPRE_ParCSRHybridGetDSCGNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * dscg_num_its )
```

Retrieves the number of iterations used by the diagonally scaled solver.

2.4.3.311 HYPRE_ParCSRHybridGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRHybridGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

Retrieves the final relative residual norm.

2.4.3.312 HYPRE_ParCSRHybridGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRHybridGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_its )
```

Retrieves the total number of iterations.

2.4.3.313 HYPRE_ParCSRHybridGetPCGNumIterations()

```
HYPRE_Int HYPRE_ParCSRHybridGetPCGNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * pcg_num_its )
```

Retrieves the number of iterations used by the AMG preconditioned solver.

2.4.3.314 HYPRE_ParCSRHybridGetRecomputeResidual()

```
HYPRE_Int HYPRE_ParCSRHybridGetRecomputeResidual (
            HYPRE_Solver solver,
            HYPRE_Int * recompute_residual )
```

(Optional) Get recompute residual option.

2.4.3.315 HYPRE_ParCSRHybridGetRecomputeResidualP()

```
HYPRE_Int HYPRE_ParCSRHybridGetRecomputeResidualP (
            HYPRE_Solver solver,
            HYPRE_Int * recompute_residual_p )
```

(Optional) Get recompute residual period option.

2.4.3.316 HYPRE_ParCSRHybridSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRHybridSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

Set the absolute convergence tolerance for the Krylov solver. The default is 0.

2.4.3.317 HYPRE_ParCSRHybridSetAggNumLevels()

```
HYPRE_Int HYPRE_ParCSRHybridSetAggNumLevels (
            HYPRE_Solver solver,
            HYPRE_Int agg_num_levels )
```

(Optional) Defines the number of levels of aggressive coarsening, starting with the finest level. The default is 0, i.e. no aggressive coarsening.

2.4.3.318 HYPRE_ParCSRHybridSetCoarsenType()

```
HYPRE_Int HYPRE_ParCSRHybridSetCoarsenType (
            HYPRE_Solver solver,
            HYPRE_Int coarsen_type )
```

(Optional) Defines which parallel coarsening algorithm is used. There are the following options for coarsen_type:

\begin{tabular}{|c|l|} \hline 0 & CLJP-coarsening (a parallel coarsening algorithm using independent sets). \ 1 & classical Ruge-Stueben coarsening on each processor, no boundary treatment \ 3 & classical Ruge-Stueben coarsening on each processor, followed by a third \ & pass, which adds coarse points on the boundaries \ 6 & Falgout coarsening (uses 1 first, followed by CLJP using the interior coarse \ & points generated by 1 as its first independent set) \ 7 & CLJP-coarsening (using a fixed random vector, for debugging purposes only) \ 8 & PMIS-coarsening (a parallel coarsening algorithm using independent sets \ & with lower complexities than CLJP, might also lead to slower convergence) \ 9 & PMIS-coarsening (using a fixed random vector, for debugging purposes only) \ 10 & HMIS-coarsening (uses one pass Ruge-Stueben on each processor independently, \ & followed by PMIS using the interior C-points as its first independent set) \ 11 & one-pass Ruge-Stueben coarsening on each processor, no boundary treatment \ \hline \end{tabular}

The default is 10.

2.4.3.319 HYPRE_ParCSRHybridSetConvergenceTol()

```
HYPRE_Int HYPRE_ParCSRHybridSetConvergenceTol (
            HYPRE_Solver solver,
            HYPRE_Real cf_tol )
```

Set the desired convergence factor

2.4.3.320 HYPRE_ParCSRHybridSetCycleNumSweeps()

```
HYPRE_Int HYPRE_ParCSRHybridSetCycleNumSweeps (
            HYPRE_Solver solver,
            HYPRE_Int num_sweeps,
            HYPRE_Int k )
```

(Optional) Sets the number of sweeps at a specified cycle. There are the following options for k:

\begin{tabular}{|l|l|} \hline the down cycle & if k=1 \ the up cycle & if k=2 \ the coarsest level & if k=3.\ \hline \end{tabular}

2.4.3.321 HYPRE_ParCSRHybridSetCycleRelaxType()

```
HYPRE_Int HYPRE_ParCSRHybridSetCycleRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int relax_type,
            HYPRE_Int k )
```

(Optional) Defines the smoother at a given cycle. For options of relax_type see description of HYPRE_BoomerAMGSetRelaxType). Options for k are

\begin{tabular}{|l|l|} \hline the down cycle & if k=1 \ the up cycle & if k=2 \ the coarsest level & if k=3. \ \hline \end{tabular}

2.4.3.322 HYPRE_ParCSRHybridSetCycleType()

```
HYPRE_Int HYPRE_ParCSRHybridSetCycleType (
            HYPRE_Solver solver,
            HYPRE_Int cycle_type )
```

(Optional) Defines the type of cycle. For a V-cycle, set cycle_type to 1, for a W-cycle set cycle_type to 2. The default is 1.

2.4.3.323 HYPRE_ParCSRHybridSetDofFunc()

```
HYPRE_Int HYPRE_ParCSRHybridSetDofFunc (
            HYPRE_Solver solver,
            HYPRE_Int * dof_func )
```

(Optional) Sets the mapping that assigns the function to each variable, if using the systems version. If no assignment is made and the number of functions is k > 1, the mapping generated is (0,1,...,k-1,0,1,...,k-1,...).

2.4.3.324 HYPRE_ParCSRHybridSetDSCGMaxIter()

```
HYPRE_Int HYPRE_ParCSRHybridSetDSCGMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int dscg_max_its )
```

Set the maximal number of iterations for the diagonally preconditioned solver

2.4.3.325 HYPRE_ParCSRHybridSetGridRelaxPoints()

```
HYPRE_Int HYPRE_ParCSRHybridSetGridRelaxPoints (
            HYPRE_Solver solver,
            HYPRE_Int ** grid_relax_points )
```

2.4.3.326 HYPRE_ParCSRHybridSetGridRelaxType()

```
HYPRE_Int HYPRE_ParCSRHybridSetGridRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int * grid_relax_type )
```

2.4.3.327 HYPRE_ParCSRHybridSetInterpType()

```
HYPRE_Int HYPRE_ParCSRHybridSetInterpType (
            HYPRE_Solver solver,
            HYPRE_Int interp_type )
```

2.4.3.328 HYPRE_ParCSRHybridSetKDim()

```
HYPRE_Int HYPRE_ParCSRHybridSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

Set the Krylov dimension for restarted GMRES. The default is 5.

2.4.3.329 HYPRE_ParCSRHybridSetKeepTranspose()

```
HYPRE_Int HYPRE_ParCSRHybridSetKeepTranspose (
            HYPRE_Solver solver,
            HYPRE_Int keepT )
```

(Optional) Sets whether to store local transposed interpolation The default is 0 (don't store).

2.4.3.330 HYPRE_ParCSRHybridSetLevelOuterWt()

```
HYPRE_Int HYPRE_ParCSRHybridSetLevelOuterWt (
            HYPRE_Solver solver,
            HYPRE_Real outer_wt,
            HYPRE_Int level )
```

(Optional) Defines the outer relaxation weight for hybrid SOR or SSOR on the user defined level. Note that the finest level is denoted 0, the next coarser level 1, etc. For nonpositive omega, the parameter is determined on the given level as described for HYPRE_BoomerAMGSetOuterWt. The default is 1.

2.4.3.331 HYPRE_ParCSRHybridSetLevelRelaxWt()

```
HYPRE_Int HYPRE_ParCSRHybridSetLevelRelaxWt (
            HYPRE_Solver solver,
            HYPRE_Real relax_wt,
            HYPRE_Int level )
```

(Optional) Defines the relaxation weight for smoothed Jacobi and hybrid SOR on the user defined level. Note that the finest level is denoted 0, the next coarser level 1, etc. For nonpositive relax_weight, the parameter is determined on the given level as described for HYPRE_BoomerAMGSetRelaxWt. The default is 1.

2.4.3.332 HYPRE_ParCSRHybridSetLogging()

```
HYPRE_Int HYPRE_ParCSRHybridSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

Set logging parameter (default: 0, no logging).

2.4.3.333 HYPRE_ParCSRHybridSetMaxCoarseSize()

```
HYPRE_Int HYPRE_ParCSRHybridSetMaxCoarseSize (
            HYPRE_Solver solver,
            HYPRE_Int max_coarse_size )
```

(Optional) Defines the maximal coarse grid size. The default is 9.

2.4.3.334 HYPRE_ParCSRHybridSetMaxLevels()

```
HYPRE_Int HYPRE_ParCSRHybridSetMaxLevels (
            HYPRE_Solver solver,
            HYPRE_Int max_levels )
```

(Optional) Defines the maximal number of levels used for AMG. The default is 25.

2.4.3.335 HYPRE_ParCSRHybridSetMaxRowSum()

```
HYPRE_Int HYPRE_ParCSRHybridSetMaxRowSum (
            HYPRE_Solver solver,
            HYPRE_Real max_row_sum )
```

(Optional) Sets a parameter to modify the definition of strength for diagonal dominant portions of the matrix. The default is 0.9. If max_row_sum is 1, no checking for diagonally dominant rows is performed.

2.4.3.336 HYPRE_ParCSRHybridSetMeasureType()

```
HYPRE_Int HYPRE_ParCSRHybridSetMeasureType (
            HYPRE_Solver solver,
            HYPRE_Int measure_type )
```

(Optional) Defines whether local or global measures are used.

2.4.3.337 HYPRE_ParCSRHybridSetMinCoarseSize()

```
HYPRE_Int HYPRE_ParCSRHybridSetMinCoarseSize (
            HYPRE_Solver solver,
            HYPRE_Int min_coarse_size )
```

(Optional) Defines the minimal coarse grid size. The default is 0.

2.4.3.338 HYPRE_ParCSRHybridSetNodal()

```
HYPRE_Int HYPRE_ParCSRHybridSetNodal (
            HYPRE_Solver solver,
            HYPRE_Int nodal )
```

(Optional) Sets whether to use the nodal systems version. The default is 0 (the unknown based approach).

2.4.3.339 HYPRE_ParCSRHybridSetNonGalerkinTol()

```
HYPRE_Int HYPRE_ParCSRHybridSetNonGalerkinTol (
            HYPRE_Solver solver,
            HYPRE_Int num_levels,
            HYPRE_Real * nongalerkin_tol )
```

(Optional) Sets whether to use non-Galerkin option The default is no non-Galerkin option num_levels sets the number of levels where to use it nongalerkin_tol contains the tolerances for <num_levels> levels

2.4.3.340 HYPRE_ParCSRHybridSetNumFunctions()

```
HYPRE_Int HYPRE_ParCSRHybridSetNumFunctions (
            HYPRE_Solver solver,
            HYPRE_Int num_functions )
```

(Optional) Sets the size of the system of PDEs, if using the systems version. The default is 1.

2.4.3.341 HYPRE_ParCSRHybridSetNumGridSweeps()

```
HYPRE_Int HYPRE_ParCSRHybridSetNumGridSweeps (
            HYPRE_Solver solver,
            HYPRE_Int * num_grid_sweeps )
```

2.4.3.342 HYPRE_ParCSRHybridSetNumPaths()

```
HYPRE_Int HYPRE_ParCSRHybridSetNumPaths (
            HYPRE_Solver solver,
            HYPRE_Int num_paths )
```

(Optional) Defines the degree of aggressive coarsening. The default is 1, which leads to the most aggressive coarsening. Setting num$_$paths to 2 will increase complexity somewhat, but can lead to better convergence.

2.4.3.343 HYPRE_ParCSRHybridSetNumSweeps()

```
HYPRE_Int HYPRE_ParCSRHybridSetNumSweeps (
            HYPRE_Solver solver,
            HYPRE_Int num_sweeps )
```

(Optional) Sets the number of sweeps. On the finest level, the up and the down cycle the number of sweeps are set to num_sweeps and on the coarsest level to 1. The default is 1.

2.4.3.344 HYPRE_ParCSRHybridSetOmega()

```
HYPRE_Int HYPRE_ParCSRHybridSetOmega (
            HYPRE_Solver solver,
            HYPRE_Real * omega )
```

2.4.3.345 HYPRE_ParCSRHybridSetOuterWt()

```
HYPRE_Int HYPRE_ParCSRHybridSetOuterWt (
            HYPRE_Solver solver,
            HYPRE_Real outer_wt )
```

(Optional) Defines the outer relaxation weight for hybrid SOR and SSOR on all levels.

\begin{tabular}{|l|l|} \hline omega $>$ 0 & this assigns the same outer relaxation weight omega on each level\ omega = -k & an outer relaxation weight is determined with at most k CG steps on each level \ & (this only makes sense for symmetric positive definite problems and smoothers, e.g. SSOR) \ \hline \end{tabular}

The default is 1.

2.4.3.346 HYPRE_ParCSRHybridSetPCGMaxIter()

```
HYPRE_Int HYPRE_ParCSRHybridSetPCGMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int pcg_max_its )
```

Set the maximal number of iterations for the AMG preconditioned solver

2.4.3.347 HYPRE_ParCSRHybridSetPMaxElmts()

```
HYPRE_Int HYPRE_ParCSRHybridSetPMaxElmts (
            HYPRE_Solver solver,
            HYPRE_Int P_max_elmts )
```

(Optional) Defines the maximal number of elements per row for the interpolation. The default is 0.

2.4.3.348 HYPRE_ParCSRHybridSetPrecond()

```
HYPRE_Int HYPRE_ParCSRHybridSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

Set preconditioner if wanting to use one that is not set up by the hybrid solver.

2.4.3.349 HYPRE_ParCSRHybridSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRHybridSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

Set print level (default: 0, no printing) 2 will print residual norms per iteration 10 will print AMG setup information if AMG is used 12 both Setup information and iterations.

2.4.3.350 HYPRE_ParCSRHybridSetRecomputeResidual()

```
HYPRE_Int HYPRE_ParCSRHybridSetRecomputeResidual (
            HYPRE_Solver solver,
            HYPRE_Int recompute_residual )
```

(Optional) Set recompute residual (don't rely on 3-term recurrence).

2.4.3.351 HYPRE_ParCSRHybridSetRecomputeResidualP()

```
HYPRE_Int HYPRE_ParCSRHybridSetRecomputeResidualP (
            HYPRE_Solver solver,
            HYPRE_Int recompute_residual_p )
```

(Optional) Set recompute residual period (don't rely on 3-term recurrence).

Recomputes residual after every specified number of iterations.

2.4.3.352 HYPRE_ParCSRHybridSetRelaxOrder()

```
HYPRE_Int HYPRE_ParCSRHybridSetRelaxOrder (
            HYPRE_Solver solver,
            HYPRE_Int relax_order )
```

(Optional) Defines in which order the points are relaxed. There are the following options for relax_order:

\begin{tabular}{|c|l|} \hline 0 & the points are relaxed in natural or lexicographic order on each processor \ 1 & CF-relaxation is used, i.e on the fine grid and the down cycle the coarse points are relaxed first, \ & followed by the fine points; on the up cycle the F-points are relaxed first, followed by the C-points. \ & On the coarsest level, if an iterative scheme is used, the points are relaxed in lexicographic order. \ \hline \end{tabular}

The default is 0 (CF-relaxation).

2.4.3.353 HYPRE_ParCSRHybridSetRelaxType()

```
HYPRE_Int HYPRE_ParCSRHybridSetRelaxType (
            HYPRE_Solver solver,
            HYPRE_Int relax_type )
```

(Optional) Defines the smoother to be used. It uses the given smoother on the fine grid, the up and the down cycle and sets the solver on the coarsest level to Gaussian elimination (9). The default is l1-Gauss-Seidel, forward solve on the down cycle (13) and backward solve on the up cycle (14).

There are the following options for relax_type:

\begin{tabular}{|c|l|} \hline 0 & Jacobi \ 1 & Gauss-Seidel, sequential (very slow!) \ 2 & Gauss-Seidel, interior points in parallel, boundary sequential (slow!) \ 3 & hybrid Gauss-Seidel or SOR, forward solve \ 4 & hybrid Gauss-Seidel or SOR, backward solve \ 6 & hybrid symmetric Gauss-Seidel or SSOR \ 8 & hybrid symmetric l1-Gauss-Seidel or SSOR \ 13 & l1-Gauss-Seidel, forward solve \ 14 & l1-Gauss-Seidel, backward solve \ 18 & l1-Jacobi \ 9 & Gaussian elimination (only on coarsest level) \ \hline \end{tabular}

2.4.3.354 HYPRE_ParCSRHybridSetRelaxWeight()

```
HYPRE_Int HYPRE_ParCSRHybridSetRelaxWeight (
             HYPRE_Solver solver,
             HYPRE_Real * relax_weight )
```

2.4.3.355 HYPRE_ParCSRHybridSetRelaxWt()

```
HYPRE_Int HYPRE_ParCSRHybridSetRelaxWt (
             HYPRE_Solver solver,
             HYPRE_Real relax_wt )
```

(Optional) Defines the relaxation weight for smoothed Jacobi and hybrid SOR on all levels.

\begin{tabular}{|l|l|} \hline relax_weight > 0 & this assigns the given relaxation weight on all levels \ relax_weight = 0 & the weight is determined on each level with the estimate $3 \over {4|D^{-1/2}AD^{-1/2}|}$,\ & where D is the diagonal matrix of A (this should only be used with Jacobi) \ relax_weight = -k & the relaxation weight is determined with at most k CG steps on each level \ & this should only be used for symmetric positive definite problems) \ \hline \end{tabular}

The default is 1.

2.4.3.356 HYPRE_ParCSRHybridSetRelChange()

```
HYPRE_Int HYPRE_ParCSRHybridSetRelChange (
             HYPRE_Solver solver,
             HYPRE_Int rel_change )
```

2.4.3.357 HYPRE_ParCSRHybridSetSeqThreshold()

```
HYPRE_Int HYPRE_ParCSRHybridSetSeqThreshold (
             HYPRE_Solver solver,
             HYPRE_Int seq_threshold )
```

(Optional) enables redundant coarse grid size. If the system size becomes smaller than seq_threshold, sequential AMG is used on all remaining processors. The default is 0.

2.4.3.358 HYPRE_ParCSRHybridSetSetupType()

```
HYPRE_Int HYPRE_ParCSRHybridSetSetupType (
             HYPRE_Solver solver,
             HYPRE_Int setup_type )
```

2.4.3.359 HYPRE_ParCSRHybridSetSolverType()

```
HYPRE_Int HYPRE_ParCSRHybridSetSolverType (
            HYPRE_Solver solver,
            HYPRE_Int solver_type )
```

Set the desired solver type. There are the following options: \begin{tabular}{l l} 1 & PCG (default) \ 2 & GMRES \ 3 & BiCGSTAB \end{tabular}

2.4.3.360 HYPRE_ParCSRHybridSetStopCrit()

```
HYPRE_Int HYPRE_ParCSRHybridSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.4.3.361 HYPRE_ParCSRHybridSetStrongThreshold()

```
HYPRE_Int HYPRE_ParCSRHybridSetStrongThreshold (
            HYPRE_Solver solver,
            HYPRE_Real strong_threshold )
```

(Optional) Sets AMG strength threshold. The default is 0.25. For elasticity problems, a larger strength threshold, such as 0.7 or 0.8, is often better.

2.4.3.362 HYPRE_ParCSRHybridSetTol()

```
HYPRE_Int HYPRE_ParCSRHybridSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

Set the convergence tolerance for the Krylov solver. The default is 1.e-7.

2.4.3.363 HYPRE_ParCSRHybridSetTruncFactor()

```
HYPRE_Int HYPRE_ParCSRHybridSetTruncFactor (
            HYPRE_Solver solver,
            HYPRE_Real trunc_factor )
```

(Optional) Defines a truncation factor for the interpolation. The default is 0.

2.4.3.364 HYPRE_ParCSRHybridSetTwoNorm()

```
HYPRE_Int HYPRE_ParCSRHybridSetTwoNorm (
            HYPRE_Solver solver,
            HYPRE_Int two_norm )
```

Set the type of norm for PCG.

2.4.3.365 HYPRE_ParCSRHybridSetup()

```
HYPRE_Int HYPRE_ParCSRHybridSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Setup the hybrid solver

Parameters

solver	[IN] object to be set up.
A	[IN] ParCSR matrix used to construct the solver/preconditioner.
b	Ignored by this function.
x	Ignored by this function.

2.4.3.366 HYPRE_ParCSRHybridSolve()

```
HYPRE_Int HYPRE_ParCSRHybridSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Solve linear system

Parameters

solver	[IN] solver or preconditioner object to be applied.
A	[IN] ParCSR matrix, matrix of the linear system to be solved
b	[IN] right hand side of the linear system to be solved
x	[OUT] approximated solution of the linear system to be solved

2.4.3.367 HYPRE_ParCSRLGMRESCreate()

```
HYPRE_Int HYPRE_ParCSRLGMRESCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.368 HYPRE_ParCSRLGMRESDestroy()

```
HYPRE_Int HYPRE_ParCSRLGMRESDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.369 HYPRE_ParCSRLGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRLGMRESGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.370 HYPRE_ParCSRLGMRESGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRLGMRESGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.371 HYPRE_ParCSRLGMRESGetPrecond()

```
HYPRE_Int HYPRE_ParCSRLGMRESGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.372 HYPRE_ParCSRLGMRESGetResidual()

```
HYPRE_Int HYPRE_ParCSRLGMRESGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

2.4.3.373 HYPRE_ParCSRLGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real a_tol )
```

2.4.3.374 HYPRE_ParCSRLGMRESSetAugDim()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetAugDim (
            HYPRE_Solver solver,
            HYPRE_Int aug_dim )
```

2.4.3.375 HYPRE_ParCSRLGMRESSetKDim()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetKDim (
            HYPRE_Solver solver,
            HYPRE_Int k_dim )
```

2.4.3.376 HYPRE_ParCSRLGMRESSetLogging()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.377 HYPRE_ParCSRLGMRESSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.378 HYPRE_ParCSRLGMRESSetMinIter()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetMinIter (
            HYPRE_Solver solver,
            HYPRE_Int min_iter )
```

2.4.3.379 HYPRE_ParCSRLGMRESSetPrecond()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.380 HYPRE_ParCSRLGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.381 HYPRE_ParCSRLGMRESSetTol()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.382 HYPRE_ParCSRLGMRESSetup()

```
HYPRE_Int HYPRE_ParCSRLGMRESSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.383 HYPRE_ParCSRLGMRESSolve()

```
HYPRE_Int HYPRE_ParCSRLGMRESSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.384 hypre_ParCSRMultiVectorPrint()

```
HYPRE_Int hypre_ParCSRMultiVectorPrint (
            void * x_,
            const char * fileName )
```

2.4.3.385 hypre_ParCSRMultiVectorRead()

```
void* hypre_ParCSRMultiVectorRead (
            MPI_Comm comm,
            void * ii_,
            const char * fileName )
```

2.4.3.386 HYPRE_ParCSROnProcTriSetup()

```
HYPRE_Int HYPRE_ParCSROnProcTriSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix HA,
            HYPRE_ParVector Hy,
            HYPRE_ParVector Hx )
```

2.4.3.387 HYPRE_ParCSROnProcTriSolve()

```
HYPRE_Int HYPRE_ParCSROnProcTriSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix HA,
            HYPRE_ParVector Hy,
            HYPRE_ParVector Hx )
```

2.4.3.388 HYPRE_ParCSRParaSailsCreate()

```
HYPRE_Int HYPRE_ParCSRParaSailsCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

2.4.3.389 HYPRE_ParCSRParaSailsDestroy()

```
HYPRE_Int HYPRE_ParCSRParaSailsDestroy (
            HYPRE_Solver solver )
```

2.4.3.390 HYPRE_ParCSRParaSailsSetFilter()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetFilter (
            HYPRE_Solver solver,
            HYPRE_Real filter )
```

2.4.3.391 HYPRE_ParCSRParaSailsSetLoadbal()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetLoadbal (
            HYPRE_Solver solver,
            HYPRE_Real loadbal )
```

2.4.3.392 HYPRE_ParCSRParaSailsSetLogging()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.393 HYPRE_ParCSRParaSailsSetParams()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetParams (
            HYPRE_Solver solver,
            HYPRE_Real thresh,
            HYPRE_Int nlevels )
```

2.4.3.394 HYPRE_ParCSRParaSailsSetReuse()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetReuse (
            HYPRE_Solver solver,
            HYPRE_Int reuse )
```

2.4.3.395 HYPRE_ParCSRParaSailsSetSym()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetSym (
            HYPRE_Solver solver,
            HYPRE_Int sym )
```

2.4.3.396 HYPRE_ParCSRParaSailsSetup()

```
HYPRE_Int HYPRE_ParCSRParaSailsSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.397 HYPRE_ParCSRParaSailsSolve()

```
HYPRE_Int HYPRE_ParCSRParaSailsSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.398 HYPRE_ParCSRPCGCreate()

```
HYPRE_Int HYPRE_ParCSRPCGCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a solver object.

2.4.3.399 HYPRE_ParCSRPCGDestroy()

```
HYPRE_Int HYPRE_ParCSRPCGDestroy (
            HYPRE_Solver solver )
```

Destroy a solver object.

2.4.3.400 HYPRE_ParCSRPCGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_ParCSRPCGGetFinalRelativeResidualNorm (
            HYPRE_Solver solver,
            HYPRE_Real * norm )
```

2.4.3.401 HYPRE_ParCSRPCGGetNumIterations()

```
HYPRE_Int HYPRE_ParCSRPCGGetNumIterations (
            HYPRE_Solver solver,
            HYPRE_Int * num_iterations )
```

2.4.3.402 HYPRE_ParCSRPCGGetPrecond()

```
HYPRE_Int HYPRE_ParCSRPCGGetPrecond (
            HYPRE_Solver solver,
            HYPRE_Solver * precond_data )
```

2.4.3.403 HYPRE_ParCSRPCGGetResidual()

```
HYPRE_Int HYPRE_ParCSRPCGGetResidual (
            HYPRE_Solver solver,
            HYPRE_ParVector * residual )
```

Returns the residual.

2.4.3.404 HYPRE_ParCSRPCGSetAbsoluteTol()

```
HYPRE_Int HYPRE_ParCSRPCGSetAbsoluteTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.405 HYPRE_ParCSRPCGSetLogging()

```
HYPRE_Int HYPRE_ParCSRPCGSetLogging (
            HYPRE_Solver solver,
            HYPRE_Int logging )
```

2.4.3.406 HYPRE_ParCSRPCGSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRPCGSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

2.4.3.407 HYPRE_ParCSRPCGSetPrecond()

```
HYPRE_Int HYPRE_ParCSRPCGSetPrecond (
            HYPRE_Solver solver,
            HYPRE_PtrToParSolverFcn precond,
            HYPRE_PtrToParSolverFcn precond_setup,
            HYPRE_Solver precond_solver )
```

2.4.3.408 HYPRE_ParCSRPCGSetPrintLevel()

```
HYPRE_Int HYPRE_ParCSRPCGSetPrintLevel (
            HYPRE_Solver solver,
            HYPRE_Int print_level )
```

2.4.3.409 HYPRE_ParCSRPCGSetRelChange()

```
HYPRE_Int HYPRE_ParCSRPCGSetRelChange (
            HYPRE_Solver solver,
            HYPRE_Int rel_change )
```

2.4.3.410 HYPRE_ParCSRPCGSetStopCrit()

```
HYPRE_Int HYPRE_ParCSRPCGSetStopCrit (
            HYPRE_Solver solver,
            HYPRE_Int stop_crit )
```

2.4.3.411 HYPRE_ParCSRPCGSetTol()

```
HYPRE_Int HYPRE_ParCSRPCGSetTol (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

2.4.3.412 HYPRE_ParCSRPCGSetTwoNorm()

```
HYPRE_Int HYPRE_ParCSRPCGSetTwoNorm (
            HYPRE_Solver solver,
            HYPRE_Int two_norm )
```

2.4.3.413 HYPRE_ParCSRPCGSetup()

```
HYPRE_Int HYPRE_ParCSRPCGSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.414 HYPRE_ParCSRPCGSolve()

```
HYPRE_Int HYPRE_ParCSRPCGSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.415 HYPRE_ParCSRPilutCreate()

```
HYPRE_Int HYPRE_ParCSRPilutCreate (
            MPI_Comm comm,
            HYPRE_Solver * solver )
```

Create a preconditioner object.

2.4.3.416 HYPRE_ParCSRPilutDestroy()

```
HYPRE_Int HYPRE_ParCSRPilutDestroy (
            HYPRE_Solver solver )
```

Destroy a preconditioner object.

2.4.3.417 HYPRE_ParCSRPilutSetDropTolerance()

```
HYPRE_Int HYPRE_ParCSRPilutSetDropTolerance (
            HYPRE_Solver solver,
            HYPRE_Real tol )
```

(Optional)

2.4.3.418 HYPRE_ParCSRPilutSetFactorRowSize()

```
HYPRE_Int HYPRE_ParCSRPilutSetFactorRowSize (
            HYPRE_Solver solver,
            HYPRE_Int size )
```

(Optional)

2.4.3.419 HYPRE_ParCSRPilutSetMaxIter()

```
HYPRE_Int HYPRE_ParCSRPilutSetMaxIter (
            HYPRE_Solver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.4.3.420 HYPRE_ParCSRPilutSetup()

```
HYPRE_Int HYPRE_ParCSRPilutSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.421 HYPRE_ParCSRPilutSolve()

```
HYPRE_Int HYPRE_ParCSRPilutSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

Precondition the system.

2.4.3.422 HYPRE_ParCSRSetupInterpreter()

```
HYPRE_Int HYPRE_ParCSRSetupInterpreter (
            mv_InterfaceInterpreter * i )
```

Load interface interpreter. Vector part loaded with hypre_ParKrylov functions and multivector part loaded with mv_↩
TempMultiVector functions.

2.4.3.423 HYPRE_ParCSRSetupMatvec()

```
HYPRE_Int HYPRE_ParCSRSetupMatvec (
            HYPRE_MatvecFunctions * mv )
```

Load Matvec interpreter with hypre_ParKrylov functions.

2.4.3.424 HYPRE_SchwarzCreate()

```
HYPRE_Int HYPRE_SchwarzCreate (
            HYPRE_Solver * solver )
```

2.4.3.425 HYPRE_SchwarzDestroy()

```
HYPRE_Int HYPRE_SchwarzDestroy (
            HYPRE_Solver solver )
```

2.4.3.426 HYPRE_SchwarzSetDofFunc()

```
HYPRE_Int HYPRE_SchwarzSetDofFunc (
            HYPRE_Solver solver,
            HYPRE_Int * dof_func )
```

2.4.3.427 HYPRE_SchwarzSetDomainStructure()

```
HYPRE_Int HYPRE_SchwarzSetDomainStructure (
            HYPRE_Solver solver,
            HYPRE_CSRMatrix domain_structure )
```

2.4.3.428 HYPRE_SchwarzSetDomainType()

```
HYPRE_Int HYPRE_SchwarzSetDomainType (
            HYPRE_Solver solver,
            HYPRE_Int domain_type )
```

2.4.3.429 HYPRE_SchwarzSetNonSymm()

```
HYPRE_Int HYPRE_SchwarzSetNonSymm (
            HYPRE_Solver solver,
            HYPRE_Int use_nonsymm )
```

2.4.3.430 HYPRE_SchwarzSetNumFunctions()

```
HYPRE_Int HYPRE_SchwarzSetNumFunctions (
            HYPRE_Solver solver,
            HYPRE_Int num_functions )
```

2.4.3.431 HYPRE_SchwarzSetOverlap()

```
HYPRE_Int HYPRE_SchwarzSetOverlap (
            HYPRE_Solver solver,
            HYPRE_Int overlap )
```

2.4.3.432 HYPRE_SchwarzSetRelaxWeight()

```
HYPRE_Int HYPRE_SchwarzSetRelaxWeight (
            HYPRE_Solver solver,
            HYPRE_Real relax_weight )
```

2.4.3.433 HYPRE_SchwarzSetup()

```
HYPRE_Int HYPRE_SchwarzSetup (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.4.3.434 HYPRE_SchwarzSetVariant()

```
HYPRE_Int HYPRE_SchwarzSetVariant (
            HYPRE_Solver solver,
            HYPRE_Int variant )
```

2.4.3.435 HYPRE_SchwarzSolve()

```
HYPRE_Int HYPRE_SchwarzSolve (
            HYPRE_Solver solver,
            HYPRE_ParCSRMatrix A,
            HYPRE_ParVector b,
            HYPRE_ParVector x )
```

2.5 HYPRE_sstruct_ls.h File Reference

Functions

SStruct SysPFMG Solver

SysPFMG is a semicoarsening multigrid solver similar to PFMG, but for systems of PDEs. For periodic problems, users should try to set the grid size in periodic dimensions to be as close to a power-of-two as possible (for more details, see \Ref{Struct PFMG Solver}).

- HYPRE_Int HYPRE_SStructSysPFMGCreate (MPI_Comm comm, HYPRE_SStructSolver ∗solver)
- HYPRE_Int HYPRE_SStructSysPFMGDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSysPFMGSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructSysPFMGSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructSysPFMGSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructSysPFMGSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructSysPFMGSetRelChange (HYPRE_SStructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_SStructSysPFMGSetZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSysPFMGSetNonZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSysPFMGSetRelaxType (HYPRE_SStructSolver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_SStructSysPFMGSetJacobiWeight (HYPRE_SStructSolver solver, HYPRE_Real weight)

- HYPRE_Int HYPRE_SStructSysPFMGSetNumPreRelax (HYPRE_SStructSolver solver, HYPRE_Int num_pre_relax)
- HYPRE_Int HYPRE_SStructSysPFMGSetNumPostRelax (HYPRE_SStructSolver solver, HYPRE_Int num_post_relax)
- HYPRE_Int HYPRE_SStructSysPFMGSetSkipRelax (HYPRE_SStructSolver solver, HYPRE_Int skip_relax)
- HYPRE_Int HYPRE_SStructSysPFMGSetDxyz (HYPRE_SStructSolver solver, HYPRE_Real ∗dxyz)
- HYPRE_Int HYPRE_SStructSysPFMGSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructSysPFMGSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_SStructSysPFMGGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_SStructSysPFMGGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real ∗norm)

SStruct FAC Solver

- HYPRE_Int HYPRE_SStructFACCreate (MPI_Comm comm, HYPRE_SStructSolver ∗solver)
- HYPRE_Int HYPRE_SStructFACDestroy2 (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructFACAMR_RAP (HYPRE_SStructMatrix A, HYPRE_Int(∗rfactors)[HYPRE_MAXDIM], HYPRE_SStructMatrix ∗fac_A)
- HYPRE_Int HYPRE_SStructFACSetup2 (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructFACSolve3 (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructFACSetPLevels (HYPRE_SStructSolver solver, HYPRE_Int nparts, HYPRE_Int ∗plevels)
- HYPRE_Int HYPRE_SStructFACSetPRefinements (HYPRE_SStructSolver solver, HYPRE_Int nparts, HYPRE_Int(∗rfactors)[HYPRE_MAXDIM])
- HYPRE_Int HYPRE_SStructFACZeroCFSten (HYPRE_SStructMatrix A, HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int rfactors[HYPRE_MAXDIM])
- HYPRE_Int HYPRE_SStructFACZeroFCSten (HYPRE_SStructMatrix A, HYPRE_SStructGrid grid, HYPRE_Int part)
- HYPRE_Int HYPRE_SStructFACZeroAMRMatrixData (HYPRE_SStructMatrix A, HYPRE_Int part_crse, HYPRE_Int rfactors[HYPRE_MAXDIM])
- HYPRE_Int HYPRE_SStructFACZeroAMRVectorData (HYPRE_SStructVector b, HYPRE_Int ∗plevels, HYPRE_Int(∗rfactors)[HYPRE_MAXDIM])
- HYPRE_Int HYPRE_SStructFACSetMaxLevels (HYPRE_SStructSolver solver, HYPRE_Int max_levels)
- HYPRE_Int HYPRE_SStructFACSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructFACSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructFACSetRelChange (HYPRE_SStructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_SStructFACSetZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructFACSetNonZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructFACSetRelaxType (HYPRE_SStructSolver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_SStructFACSetJacobiWeight (HYPRE_SStructSolver solver, HYPRE_Real weight)
- HYPRE_Int HYPRE_SStructFACSetNumPreRelax (HYPRE_SStructSolver solver, HYPRE_Int num_pre_relax)
- HYPRE_Int HYPRE_SStructFACSetNumPostRelax (HYPRE_SStructSolver solver, HYPRE_Int num_post_relax)
- HYPRE_Int HYPRE_SStructFACSetCoarseSolverType (HYPRE_SStructSolver solver, HYPRE_Int csolver_type)
- HYPRE_Int HYPRE_SStructFACSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructFACGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_SStructFACGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real ∗norm)

SStruct Maxwell Solver

- HYPRE_Int HYPRE_SStructMaxwellCreate (MPI_Comm comm, HYPRE_SStructSolver *solver)
- HYPRE_Int HYPRE_SStructMaxwellDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructMaxwellSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructMaxwellSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructMaxwellSolve2 (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructMaxwellSetGrad (HYPRE_SStructSolver solver, HYPRE_ParCSRMatrix T)
- HYPRE_Int HYPRE_SStructMaxwellSetRfactors (HYPRE_SStructSolver solver, HYPRE_Int rfactors[HYPRE↩_MAXDIM])
- HYPRE_Int HYPRE_SStructMaxwellPhysBdy (HYPRE_SStructGrid *grid_l, HYPRE_Int num_levels, HYPRE_Int rfactors[HYPRE_MAXDIM], HYPRE_Int ***BdryRanks_ptr, HYPRE_Int **BdryRanksCnt_↩ptr)
- HYPRE_Int HYPRE_SStructMaxwellEliminateRowsCols (HYPRE_ParCSRMatrix parA, HYPRE_Int nrows, HYPRE_Int *rows)
- HYPRE_Int HYPRE_SStructMaxwellZeroVector (HYPRE_ParVector b, HYPRE_Int *rows, HYPRE_Int nrows)
- HYPRE_Int HYPRE_SStructMaxwellSetSetConstantCoef (HYPRE_SStructSolver solver, HYPRE_Int flag)
- HYPRE_Int HYPRE_SStructMaxwellGrad (HYPRE_SStructGrid grid, HYPRE_ParCSRMatrix *T)
- HYPRE_Int HYPRE_SStructMaxwellSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructMaxwellSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructMaxwellSetRelChange (HYPRE_SStructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_SStructMaxwellSetNumPreRelax (HYPRE_SStructSolver solver, HYPRE_Int num_pre↩_relax)
- HYPRE_Int HYPRE_SStructMaxwellSetNumPostRelax (HYPRE_SStructSolver solver, HYPRE_Int num_↩post_relax)
- HYPRE_Int HYPRE_SStructMaxwellSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructMaxwellGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int *num_↩iterations)
- HYPRE_Int HYPRE_SStructMaxwellGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE↩_Real *norm)

SStruct PCG Solver

These routines should be used in conjunction with the generic interface in \Ref{PCG Solver}.

- HYPRE_Int HYPRE_SStructPCGCreate (MPI_Comm comm, HYPRE_SStructSolver *solver)
- HYPRE_Int HYPRE_SStructPCGDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructPCGSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructPCGSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructPCGSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructPCGSetAbsoluteTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructPCGSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructPCGSetTwoNorm (HYPRE_SStructSolver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_SStructPCGSetRelChange (HYPRE_SStructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_SStructPCGSetPrecond (HYPRE_SStructSolver solver, HYPRE_PtrToSStructSolverFcn precond, HYPRE_PtrToSStructSolverFcn precond_setup, void *precond_solver)
- HYPRE_Int HYPRE_SStructPCGSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructPCGSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_SStructPCGGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int *num_↩iterations)
- HYPRE_Int HYPRE_SStructPCGGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_↩Real *norm)
- HYPRE_Int HYPRE_SStructPCGGetResidual (HYPRE_SStructSolver solver, void **residual)
- HYPRE_Int HYPRE_SStructDiagScaleSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector y, HYPRE_SStructVector x)

- HYPRE_Int HYPRE_SStructDiagScale (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector y, HYPRE_SStructVector x)

SStruct GMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{GMRES Solver}.

- HYPRE_Int HYPRE_SStructGMRESCreate (MPI_Comm comm, HYPRE_SStructSolver *solver)
- HYPRE_Int HYPRE_SStructGMRESDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructGMRESSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructGMRESSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructGMRESSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructGMRESSetAbsoluteTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructGMRESSetMinIter (HYPRE_SStructSolver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_SStructGMRESSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructGMRESSetKDim (HYPRE_SStructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_SStructGMRESSetStopCrit (HYPRE_SStructSolver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_SStructGMRESSetPrecond (HYPRE_SStructSolver solver, HYPRE_PtrToSStructSolverFcn precond, HYPRE_PtrToSStructSolverFcn precond_setup, void *precond_solver)
- HYPRE_Int HYPRE_SStructGMRESSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructGMRESSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_SStructGMRESGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int *num_↩ iterations)
- HYPRE_Int HYPRE_SStructGMRESGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE↩ _Real *norm)
- HYPRE_Int HYPRE_SStructGMRESGetResidual (HYPRE_SStructSolver solver, void **residual)

SStruct FlexGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{FlexGMRES Solver}.

- HYPRE_Int HYPRE_SStructFlexGMRESCreate (MPI_Comm comm, HYPRE_SStructSolver *solver)
- HYPRE_Int HYPRE_SStructFlexGMRESDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructFlexGMRESSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructFlexGMRESSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructFlexGMRESSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructFlexGMRESSetAbsoluteTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructFlexGMRESSetMinIter (HYPRE_SStructSolver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_SStructFlexGMRESSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructFlexGMRESSetKDim (HYPRE_SStructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_SStructFlexGMRESSetPrecond (HYPRE_SStructSolver solver, HYPRE_PtrToSStructSolverFcn precond, HYPRE_PtrToSStructSolverFcn precond_setup, void *precond_solver)
- HYPRE_Int HYPRE_SStructFlexGMRESSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructFlexGMRESSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_SStructFlexGMRESGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int *num_iterations)
- HYPRE_Int HYPRE_SStructFlexGMRESGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real *norm)
- HYPRE_Int HYPRE_SStructFlexGMRESGetResidual (HYPRE_SStructSolver solver, void **residual)
- HYPRE_Int HYPRE_SStructFlexGMRESSetModifyPC (HYPRE_SStructSolver solver, HYPRE_PtrToModifyPCFcn modify_pc)

SStruct LGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{LGMRES Solver}.

- HYPRE_Int HYPRE_SStructLGMRESCreate (MPI_Comm comm, HYPRE_SStructSolver ∗solver)
- HYPRE_Int HYPRE_SStructLGMRESDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructLGMRESSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructLGMRESSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructLGMRESSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructLGMRESSetAbsoluteTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructLGMRESSetMinIter (HYPRE_SStructSolver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_SStructLGMRESSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructLGMRESSetKDim (HYPRE_SStructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_SStructLGMRESSetAugDim (HYPRE_SStructSolver solver, HYPRE_Int aug_dim)
- HYPRE_Int HYPRE_SStructLGMRESSetPrecond (HYPRE_SStructSolver solver, HYPRE_PtrToSStructSolverFcn precond, HYPRE_PtrToSStructSolverFcn precond_setup, void ∗precond_solver)
- HYPRE_Int HYPRE_SStructLGMRESSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructLGMRESSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_SStructLGMRESGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int ∗num_↩ iterations)
- HYPRE_Int HYPRE_SStructLGMRESGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_SStructLGMRESGetResidual (HYPRE_SStructSolver solver, void ∗∗residual)

SStruct BiCGSTAB Solver

These routines should be used in conjunction with the generic interface in \Ref{BiCGSTAB Solver}.

- HYPRE_Int HYPRE_SStructBiCGSTABCreate (MPI_Comm comm, HYPRE_SStructSolver ∗solver)
- HYPRE_Int HYPRE_SStructBiCGSTABDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructBiCGSTABSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructBiCGSTABSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructBiCGSTABSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructBiCGSTABSetAbsoluteTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructBiCGSTABSetMinIter (HYPRE_SStructSolver solver, HYPRE_Int min_iter)
- HYPRE_Int HYPRE_SStructBiCGSTABSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructBiCGSTABSetStopCrit (HYPRE_SStructSolver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_SStructBiCGSTABSetPrecond (HYPRE_SStructSolver solver, HYPRE_PtrToSStructSolverFcn precond, HYPRE_PtrToSStructSolverFcn precond_setup, void ∗precond_solver)
- HYPRE_Int HYPRE_SStructBiCGSTABSetLogging (HYPRE_SStructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_SStructBiCGSTABSetPrintLevel (HYPRE_SStructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_SStructBiCGSTABGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int ∗num↩ _iterations)
- HYPRE_Int HYPRE_SStructBiCGSTABGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_SStructBiCGSTABGetResidual (HYPRE_SStructSolver solver, void ∗∗residual)

SStruct LOBPCG Eigensolver

These routines should be used in conjunction with the generic interface in \Ref{LOBPCG Eigensolver}.

- HYPRE_Int HYPRE_SStructSetupInterpreter (mv_InterfaceInterpreter ∗i)
- HYPRE_Int HYPRE_SStructSetupMatvec (HYPRE_MatvecFunctions ∗mv)
- HYPRE_Int hypre_SStructPVectorSetRandomValues (hypre_SStructPVector ∗pvector, HYPRE_Int seed)
- HYPRE_Int hypre_SStructVectorSetRandomValues (hypre_SStructVector ∗vector, HYPRE_Int seed)
- HYPRE_Int hypre_SStructSetRandomValues (void ∗v, HYPRE_Int seed)

SStruct Solvers

- #define HYPRE_MODIFYPC
- #define HYPRE_SOLVER_STRUCT
- typedef struct hypre_SStructSolver_struct ∗ HYPRE_SStructSolver
- typedef HYPRE_Int(∗ HYPRE_PtrToSStructSolverFcn) (HYPRE_SStructSolver, HYPRE_SStructMatrix, HYPRE_SStructVector, HYPRE_SStructVector)
- typedef struct hypre_Solver_struct ∗ HYPRE_Solver
- typedef HYPRE_Int(∗ HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)

SStruct Split Solver

- #define HYPRE_PFMG 10
- #define HYPRE_SMG 11
- #define HYPRE_Jacobi 17
- HYPRE_Int HYPRE_SStructSplitCreate (MPI_Comm comm, HYPRE_SStructSolver ∗solver)
- HYPRE_Int HYPRE_SStructSplitDestroy (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSplitSetup (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructSplitSolve (HYPRE_SStructSolver solver, HYPRE_SStructMatrix A, HYPRE_SStructVector b, HYPRE_SStructVector x)
- HYPRE_Int HYPRE_SStructSplitSetTol (HYPRE_SStructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_SStructSplitSetMaxIter (HYPRE_SStructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_SStructSplitSetZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSplitSetNonZeroGuess (HYPRE_SStructSolver solver)
- HYPRE_Int HYPRE_SStructSplitSetStructSolver (HYPRE_SStructSolver solver, HYPRE_Int ssolver)
- HYPRE_Int HYPRE_SStructSplitGetNumIterations (HYPRE_SStructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_SStructSplitGetFinalRelativeResidualNorm (HYPRE_SStructSolver solver, HYPRE_Real ∗norm)

2.5.1 Macro Definition Documentation

2.5.1.1 HYPRE_Jacobi

```
#define HYPRE_Jacobi 17
```

2.5.1.2 HYPRE_MODIFYPC

```
#define HYPRE_MODIFYPC
```

2.5.1.3 HYPRE_PFMG

```
#define HYPRE_PFMG 10
```

2.5.1.4 HYPRE_SMG

```
#define HYPRE_SMG 11
```

2.5.1.5 HYPRE_SOLVER_STRUCT

```
#define HYPRE_SOLVER_STRUCT
```

2.5.2 Typedef Documentation

2.5.2.1 HYPRE_PtrToModifyPCFcn

```
typedef HYPRE_Int(* HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)
```

2.5.2.2 HYPRE_PtrToSStructSolverFcn

```
typedef HYPRE_Int(* HYPRE_PtrToSStructSolverFcn) (HYPRE_SStructSolver, HYPRE_SStructMatrix, HYPRE_SStructVector,
HYPRE_SStructVector)
```

2.5.2.3 HYPRE_Solver

```
typedef struct hypre_Solver_struct* HYPRE_Solver
```

2.5.2.4 HYPRE_SStructSolver

```
typedef struct hypre_SStructSolver_struct* HYPRE_SStructSolver
```

The solver object.

2.5.3 Function Documentation

2.5.3.1 HYPRE_SStructBiCGSTABCreate()

```
HYPRE_Int HYPRE_SStructBiCGSTABCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.2 HYPRE_SStructBiCGSTABDestroy()

```
HYPRE_Int HYPRE_SStructBiCGSTABDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.3 HYPRE_SStructBiCGSTABGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructBiCGSTABGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

2.5.3.4 HYPRE_SStructBiCGSTABGetNumIterations()

```
HYPRE_Int HYPRE_SStructBiCGSTABGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

2.5.3.5 HYPRE_SStructBiCGSTABGetResidual()

```
HYPRE_Int HYPRE_SStructBiCGSTABGetResidual (
            HYPRE_SStructSolver solver,
            void ** residual )
```

2.5.3.6 HYPRE_SStructBiCGSTABSetAbsoluteTol()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetAbsoluteTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.7 HYPRE_SStructBiCGSTABSetLogging()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

2.5.3.8 HYPRE_SStructBiCGSTABSetMaxIter()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

2.5.3.9 HYPRE_SStructBiCGSTABSetMinIter()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetMinIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int min_iter )
```

2.5.3.10 HYPRE_SStructBiCGSTABSetPrecond()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetPrecond (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToSStructSolverFcn precond,
            HYPRE_PtrToSStructSolverFcn precond_setup,
            void * precond_solver )
```

2.5.3.11 HYPRE_SStructBiCGSTABSetPrintLevel()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int level )
```

2.5.3.12 HYPRE_SStructBiCGSTABSetStopCrit()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetStopCrit (
            HYPRE_SStructSolver solver,
            HYPRE_Int stop_crit )
```

2.5.3.13 HYPRE_SStructBiCGSTABSetTol()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.14 HYPRE_SStructBiCGSTABSetup()

```
HYPRE_Int HYPRE_SStructBiCGSTABSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.15 HYPRE_SStructBiCGSTABSolve()

```
HYPRE_Int HYPRE_SStructBiCGSTABSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.16 HYPRE_SStructDiagScale()

```
HYPRE_Int HYPRE_SStructDiagScale (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector y,
            HYPRE_SStructVector x )
```

Solve routine for diagonal preconditioning.

2.5.3.17 HYPRE_SStructDiagScaleSetup()

```
HYPRE_Int HYPRE_SStructDiagScaleSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector y,
            HYPRE_SStructVector x )
```

Setup routine for diagonal preconditioning.

2.5.3.18 HYPRE_SStructFACAMR_RAP()

```
HYPRE_Int HYPRE_SStructFACAMR_RAP (
            HYPRE_SStructMatrix A,
            HYPRE_Int(*) rfactors[HYPRE_MAXDIM],
            HYPRE_SStructMatrix * fac_A )
```

Re-distribute the composite matrix so that the amr hierachy is approximately nested. Coarse underlying operators are also formed.

2.5.3.19 HYPRE_SStructFACCreate()

```
HYPRE_Int HYPRE_SStructFACCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.20 HYPRE_SStructFACDestroy2()

```
HYPRE_Int HYPRE_SStructFACDestroy2 (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.21 HYPRE_SStructFACGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructFACGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.5.3.22 HYPRE_SStructFACGetNumIterations()

```
HYPRE_Int HYPRE_SStructFACGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.5.3.23 HYPRE_SStructFACSetCoarseSolverType()

```
HYPRE_Int HYPRE_SStructFACSetCoarseSolverType (
            HYPRE_SStructSolver solver,
            HYPRE_Int csolver_type )
```

(Optional) Set coarsest solver type.

Current solver types set by {\tt csolver_type} are:

\begin{tabular}{l – }l} 1 & SysPFMG-PCG (default) \ 2 & SysPFMG \ \end{tabular}

2.5.3.24 HYPRE_SStructFACSetJacobiWeight()

```
HYPRE_Int HYPRE_SStructFACSetJacobiWeight (
            HYPRE_SStructSolver solver,
            HYPRE_Real weight )
```

(Optional) Set Jacobi weight if weighted Jacobi is used.

2.5.3.25 HYPRE_SStructFACSetLogging()

```
HYPRE_Int HYPRE_SStructFACSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.5.3.26 HYPRE_SStructFACSetMaxIter()

```
HYPRE_Int HYPRE_SStructFACSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.5.3.27 HYPRE_SStructFACSetMaxLevels()

```
HYPRE_Int HYPRE_SStructFACSetMaxLevels (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_levels )
```

(Optional) Set maximum number of FAC levels.

2.5.3.28 HYPRE_SStructFACSetNonZeroGuess()

```
HYPRE_Int HYPRE_SStructFACSetNonZeroGuess (
            HYPRE_SStructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.5.3.29 HYPRE_SStructFACSetNumPostRelax()

```
HYPRE_Int HYPRE_SStructFACSetNumPostRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_post_relax )
```

(Optional) Set number of relaxation sweeps after coarse-grid correction.

2.5.3.30 HYPRE_SStructFACSetNumPreRelax()

```
HYPRE_Int HYPRE_SStructFACSetNumPreRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_pre_relax )
```

(Optional) Set number of relaxation sweeps before coarse-grid correction.

2.5.3.31 HYPRE_SStructFACSetPLevels()

```
HYPRE_Int HYPRE_SStructFACSetPLevels (
            HYPRE_SStructSolver solver,
            HYPRE_Int nparts,
            HYPRE_Int * plevels )
```

Set up amr structure

2.5.3.32 HYPRE_SStructFACSetPRefinements()

```
HYPRE_Int HYPRE_SStructFACSetPRefinements (
            HYPRE_SStructSolver solver,
            HYPRE_Int nparts,
            HYPRE_Int(*) rfactors[HYPRE_MAXDIM] )
```

Set up amr refinement factors

2.5.3.33 HYPRE_SStructFACSetRelaxType()

```
HYPRE_Int HYPRE_SStructFACSetRelaxType (
            HYPRE_SStructSolver solver,
            HYPRE_Int relax_type )
```

(Optional) Set relaxation type. See \Ref{HYPRE_SStructSysPFMGSetRelaxType} for appropriate values of {\tt relax_↩
type}.

2.5.3.34 HYPRE_SStructFACSetRelChange()

```
HYPRE_Int HYPRE_SStructFACSetRelChange (
            HYPRE_SStructSolver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.5.3.35 HYPRE_SStructFACSetTol()

```
HYPRE_Int HYPRE_SStructFACSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.5.3.36 HYPRE_SStructFACSetup2()

```
HYPRE_Int HYPRE_SStructFACSetup2 (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Set up the FAC solver structure .

2.5.3.37 HYPRE_SStructFACSetZeroGuess()

```
HYPRE_Int HYPRE_SStructFACSetZeroGuess (
            HYPRE_SStructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.5.3.38 HYPRE_SStructFACSolve3()

```
HYPRE_Int HYPRE_SStructFACSolve3 (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Solve the system.

2.5.3.39 HYPRE_SStructFACZeroAMRMatrixData()

```
HYPRE_Int HYPRE_SStructFACZeroAMRMatrixData (
            HYPRE_SStructMatrix A,
            HYPRE_Int part_crse,
            HYPRE_Int rfactors[HYPRE_MAXDIM] )
```

(Optional, but user must make sure that they do this function otherwise.) Places the identity in the coarse grid matrix underlying the fine patches. Required between each pair of amr levels.

2.5.3.40 HYPRE_SStructFACZeroAMRVectorData()

```
HYPRE_Int HYPRE_SStructFACZeroAMRVectorData (
            HYPRE_SStructVector b,
            HYPRE_Int * plevels,
            HYPRE_Int(*) rfactors[HYPRE_MAXDIM] )
```

(Optional, but user must make sure that they do this function otherwise.) Places zeros in the coarse grid vector underlying the fine patches. Required between each pair of amr levels.

2.5.3.41 HYPRE_SStructFACZeroCFSten()

```
HYPRE_Int HYPRE_SStructFACZeroCFSten (
            HYPRE_SStructMatrix A,
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int rfactors[HYPRE_MAXDIM] )
```

(Optional, but user must make sure that they do this function otherwise.) Zero off the coarse level stencils reaching into a fine level grid.

2.5.3.42 HYPRE_SStructFACZeroFCSten()

```
HYPRE_Int HYPRE_SStructFACZeroFCSten (
            HYPRE_SStructMatrix A,
            HYPRE_SStructGrid grid,
            HYPRE_Int part )
```

(Optional, but user must make sure that they do this function otherwise.) Zero off the fine level stencils reaching into a coarse level grid.

2.5.3.43 HYPRE_SStructFlexGMRESCreate()

```
HYPRE_Int HYPRE_SStructFlexGMRESCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.44 HYPRE_SStructFlexGMRESDestroy()

```
HYPRE_Int HYPRE_SStructFlexGMRESDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.45 HYPRE_SStructFlexGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructFlexGMRESGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

2.5.3.46 HYPRE_SStructFlexGMRESGetNumIterations()

```
HYPRE_Int HYPRE_SStructFlexGMRESGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

2.5.3.47 HYPRE_SStructFlexGMRESGetResidual()

```
HYPRE_Int HYPRE_SStructFlexGMRESGetResidual (
            HYPRE_SStructSolver solver,
            void ** residual )
```

2.5.3.48 HYPRE_SStructFlexGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetAbsoluteTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.49 HYPRE_SStructFlexGMRESSetKDim()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetKDim (
            HYPRE_SStructSolver solver,
            HYPRE_Int k_dim )
```

2.5.3.50 HYPRE_SStructFlexGMRESSetLogging()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

2.5.3.51 HYPRE_SStructFlexGMRESSetMaxIter()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

2.5.3.52 HYPRE_SStructFlexGMRESSetMinIter()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetMinIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int min_iter )
```

2.5.3.53 HYPRE_SStructFlexGMRESSetModifyPC()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetModifyPC (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToModifyPCFcn modify_pc )
```

2.5.3.54 HYPRE_SStructFlexGMRESSetPrecond()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetPrecond (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToSStructSolverFcn precond,
            HYPRE_PtrToSStructSolverFcn precond_setup,
            void * precond_solver )
```

2.5.3.55 HYPRE_SStructFlexGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int print_level )
```

2.5.3.56 HYPRE_SStructFlexGMRESSetTol()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.57 HYPRE_SStructFlexGMRESSetup()

```
HYPRE_Int HYPRE_SStructFlexGMRESSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.58 HYPRE_SStructFlexGMRESSolve()

```
HYPRE_Int HYPRE_SStructFlexGMRESSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.59 HYPRE_SStructGMRESCreate()

```
HYPRE_Int HYPRE_SStructGMRESCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.60 HYPRE_SStructGMRESDestroy()

```
HYPRE_Int HYPRE_SStructGMRESDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.61 HYPRE_SStructGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructGMRESGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

2.5.3.62 HYPRE_SStructGMRESGetNumIterations()

```
HYPRE_Int HYPRE_SStructGMRESGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

2.5.3.63 HYPRE_SStructGMRESGetResidual()

```
HYPRE_Int HYPRE_SStructGMRESGetResidual (
            HYPRE_SStructSolver solver,
            void ** residual )
```

2.5.3.64 HYPRE_SStructGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_SStructGMRESSetAbsoluteTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.65 HYPRE_SStructGMRESSetKDim()

```
HYPRE_Int HYPRE_SStructGMRESSetKDim (
            HYPRE_SStructSolver solver,
            HYPRE_Int k_dim )
```

2.5.3.66 HYPRE_SStructGMRESSetLogging()

```
HYPRE_Int HYPRE_SStructGMRESSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

2.5.3.67 HYPRE_SStructGMRESSetMaxIter()

```
HYPRE_Int HYPRE_SStructGMRESSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

2.5.3.68 HYPRE_SStructGMRESSetMinIter()

```
HYPRE_Int HYPRE_SStructGMRESSetMinIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int min_iter )
```

2.5.3.69 HYPRE_SStructGMRESSetPrecond()

```
HYPRE_Int HYPRE_SStructGMRESSetPrecond (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToSStructSolverFcn precond,
            HYPRE_PtrToSStructSolverFcn precond_setup,
            void * precond_solver )
```

2.5.3.70 HYPRE_SStructGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_SStructGMRESSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int print_level )
```

2.5.3.71 HYPRE_SStructGMRESSetStopCrit()

```
HYPRE_Int HYPRE_SStructGMRESSetStopCrit (
            HYPRE_SStructSolver solver,
            HYPRE_Int stop_crit )
```

2.5.3.72 HYPRE_SStructGMRESSetTol()

```
HYPRE_Int HYPRE_SStructGMRESSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.73 HYPRE_SStructGMRESSetup()

```
HYPRE_Int HYPRE_SStructGMRESSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.74 HYPRE_SStructGMRESSolve()

```
HYPRE_Int HYPRE_SStructGMRESSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.75 HYPRE_SStructLGMRESCreate()

```
HYPRE_Int HYPRE_SStructLGMRESCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.76 HYPRE_SStructLGMRESDestroy()

```
HYPRE_Int HYPRE_SStructLGMRESDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.77 HYPRE_SStructLGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructLGMRESGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

2.5.3.78 HYPRE_SStructLGMRESGetNumIterations()

```
HYPRE_Int HYPRE_SStructLGMRESGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

2.5.3.79 HYPRE_SStructLGMRESGetResidual()

```
HYPRE_Int HYPRE_SStructLGMRESGetResidual (
            HYPRE_SStructSolver solver,
            void ** residual )
```

2.5.3.80 HYPRE_SStructLGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_SStructLGMRESSetAbsoluteTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.81 HYPRE_SStructLGMRESSetAugDim()

```
HYPRE_Int HYPRE_SStructLGMRESSetAugDim (
            HYPRE_SStructSolver solver,
            HYPRE_Int aug_dim )
```

2.5.3.82 HYPRE_SStructLGMRESSetKDim()

```
HYPRE_Int HYPRE_SStructLGMRESSetKDim (
            HYPRE_SStructSolver solver,
            HYPRE_Int k_dim )
```

2.5.3.83 HYPRE_SStructLGMRESSetLogging()

```
HYPRE_Int HYPRE_SStructLGMRESSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

2.5.3.84 HYPRE_SStructLGMRESSetMaxIter()

```
HYPRE_Int HYPRE_SStructLGMRESSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

2.5.3.85 HYPRE_SStructLGMRESSetMinIter()

```
HYPRE_Int HYPRE_SStructLGMRESSetMinIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int min_iter )
```

2.5.3.86 HYPRE_SStructLGMRESSetPrecond()

```
HYPRE_Int HYPRE_SStructLGMRESSetPrecond (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToSStructSolverFcn precond,
            HYPRE_PtrToSStructSolverFcn precond_setup,
            void * precond_solver )
```

2.5.3.87 HYPRE_SStructLGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_SStructLGMRESSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int print_level )
```

2.5.3.88 HYPRE_SStructLGMRESSetTol()

```
HYPRE_Int HYPRE_SStructLGMRESSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.89 HYPRE_SStructLGMRESSetup()

```
HYPRE_Int HYPRE_SStructLGMRESSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.90 HYPRE_SStructLGMRESSolve()

```
HYPRE_Int HYPRE_SStructLGMRESSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.91 HYPRE_SStructMaxwellCreate()

```
HYPRE_Int HYPRE_SStructMaxwellCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.92 HYPRE_SStructMaxwellDestroy()

```
HYPRE_Int HYPRE_SStructMaxwellDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.93 HYPRE_SStructMaxwellEliminateRowsCols()

```
HYPRE_Int HYPRE_SStructMaxwellEliminateRowsCols (
            HYPRE_ParCSRMatrix parA,
            HYPRE_Int nrows,
            HYPRE_Int * rows )
```

Eliminates the rows and cols corresponding to the physical boundary in a parcsr matrix.

2.5.3.94 HYPRE_SStructMaxwellGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructMaxwellGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.5.3.95 HYPRE_SStructMaxwellGetNumIterations()

```
HYPRE_Int HYPRE_SStructMaxwellGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.5.3.96 HYPRE_SStructMaxwellGrad()

```
HYPRE_Int HYPRE_SStructMaxwellGrad (
            HYPRE_SStructGrid grid,
            HYPRE_ParCSRMatrix * T )
```

(Optional) Creates a gradient matrix from the grid. This presupposes a particular orientation of the edge elements.

2.5.3.97 HYPRE_SStructMaxwellPhysBdy()

```
HYPRE_Int HYPRE_SStructMaxwellPhysBdy (
            HYPRE_SStructGrid * grid_l,
            HYPRE_Int num_levels,
            HYPRE_Int rfactors[HYPRE_MAXDIM],
            HYPRE_Int *** BdryRanks_ptr,
            HYPRE_Int ** BdryRanksCnt_ptr )
```

Finds the physical boundary row ranks on all levels.

2.5.3.98 HYPRE_SStructMaxwellSetGrad()

```
HYPRE_Int HYPRE_SStructMaxwellSetGrad (
            HYPRE_SStructSolver solver,
            HYPRE_ParCSRMatrix T )
```

Sets the gradient operator in the Maxwell solver.

2.5.3.99 HYPRE_SStructMaxwellSetLogging()

```
HYPRE_Int HYPRE_SStructMaxwellSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.5.3.100 HYPRE_SStructMaxwellSetMaxIter()

```
HYPRE_Int HYPRE_SStructMaxwellSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.5.3.101 HYPRE_SStructMaxwellSetNumPostRelax()

```
HYPRE_Int HYPRE_SStructMaxwellSetNumPostRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_post_relax )
```

(Optional) Set number of relaxation sweeps after coarse-grid correction.

2.5.3.102 HYPRE_SStructMaxwellSetNumPreRelax()

```
HYPRE_Int HYPRE_SStructMaxwellSetNumPreRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_pre_relax )
```

(Optional) Set number of relaxation sweeps before coarse-grid correction.

2.5.3.103 HYPRE_SStructMaxwellSetRelChange()

```
HYPRE_Int HYPRE_SStructMaxwellSetRelChange (
            HYPRE_SStructSolver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.5.3.104 HYPRE_SStructMaxwellSetRfactors()

```
HYPRE_Int HYPRE_SStructMaxwellSetRfactors (
            HYPRE_SStructSolver solver,
            HYPRE_Int rfactors[HYPRE_MAXDIM] )
```

Sets the coarsening factor.

2.5.3.105 HYPRE_SStructMaxwellSetSetConstantCoef()

```
HYPRE_Int HYPRE_SStructMaxwellSetSetConstantCoef (
            HYPRE_SStructSolver solver,
            HYPRE_Int flag )
```

(Optional) Set the constant coefficient flag- Nedelec interpolation used.

2.5.3.106 HYPRE_SStructMaxwellSetTol()

```
HYPRE_Int HYPRE_SStructMaxwellSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.5.3.107 HYPRE_SStructMaxwellSetup()

```
HYPRE_Int HYPRE_SStructMaxwellSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.5.3.108 HYPRE_SStructMaxwellSolve()

```
HYPRE_Int HYPRE_SStructMaxwellSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Solve the system. Full coupling of the augmented system used throughout the multigrid hierarchy.

2.5.3.109 HYPRE_SStructMaxwellSolve2()

```
HYPRE_Int HYPRE_SStructMaxwellSolve2 (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Solve the system. Full coupling of the augmented system used only on the finest level, i.e., the node and edge multigrid cycles are coupled only on the finest level.

2.5.3.110 HYPRE_SStructMaxwellZeroVector()

```
HYPRE_Int HYPRE_SStructMaxwellZeroVector (
            HYPRE_ParVector b,
            HYPRE_Int * rows,
            HYPRE_Int nrows )
```

Zeros the rows corresponding to the physical boundary in a par vector.

2.5.3.111 HYPRE_SStructPCGCreate()

```
HYPRE_Int HYPRE_SStructPCGCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.112 HYPRE_SStructPCGDestroy()

```
HYPRE_Int HYPRE_SStructPCGDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.113 HYPRE_SStructPCGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructPCGGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

2.5.3.114 HYPRE_SStructPCGGetNumIterations()

```
HYPRE_Int HYPRE_SStructPCGGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

2.5.3.115 HYPRE_SStructPCGGetResidual()

```
HYPRE_Int HYPRE_SStructPCGGetResidual (
            HYPRE_SStructSolver solver,
            void ** residual )
```

2.5.3.116 HYPRE_SStructPCGSetAbsoluteTol()

```
HYPRE_Int HYPRE_SStructPCGSetAbsoluteTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.117 HYPRE_SStructPCGSetLogging()

```
HYPRE_Int HYPRE_SStructPCGSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

2.5.3.118 HYPRE_SStructPCGSetMaxIter()

```
HYPRE_Int HYPRE_SStructPCGSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

2.5.3.119 HYPRE_SStructPCGSetPrecond()

```
HYPRE_Int HYPRE_SStructPCGSetPrecond (
            HYPRE_SStructSolver solver,
            HYPRE_PtrToSStructSolverFcn precond,
            HYPRE_PtrToSStructSolverFcn precond_setup,
            void * precond_solver )
```

2.5.3.120 HYPRE_SStructPCGSetPrintLevel()

```
HYPRE_Int HYPRE_SStructPCGSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int level )
```

2.5.3.121 HYPRE_SStructPCGSetRelChange()

```
HYPRE_Int HYPRE_SStructPCGSetRelChange (
            HYPRE_SStructSolver solver,
            HYPRE_Int rel_change )
```

2.5.3.122 HYPRE_SStructPCGSetTol()

```
HYPRE_Int HYPRE_SStructPCGSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

2.5.3.123 HYPRE_SStructPCGSetTwoNorm()

```
HYPRE_Int HYPRE_SStructPCGSetTwoNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Int two_norm )
```

2.5.3.124 HYPRE_SStructPCGSetup()

```
HYPRE_Int HYPRE_SStructPCGSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.125 HYPRE_SStructPCGSolve()

```
HYPRE_Int HYPRE_SStructPCGSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

2.5.3.126 hypre_SStructPVectorSetRandomValues()

```
HYPRE_Int hypre_SStructPVectorSetRandomValues (
            hypre_SStructPVector * pvector,
            HYPRE_Int seed )
```

2.5.3.127 hypre_SStructSetRandomValues()

```
HYPRE_Int hypre_SStructSetRandomValues (
            void * v,
            HYPRE_Int seed )
```

2.5.3.128 HYPRE_SStructSetupInterpreter()

```
HYPRE_Int HYPRE_SStructSetupInterpreter (
            mv_InterfaceInterpreter * i )
```

Load interface interpreter. Vector part loaded with hypre_SStructKrylov functions and multivector part loaded with mv_TempMultiVector functions.

2.5.3.129 HYPRE_SStructSetupMatvec()

```
HYPRE_Int HYPRE_SStructSetupMatvec (
            HYPRE_MatvecFunctions * mv )
```

Load Matvec interpreter with hypre_SStructKrylov functions.

2.5.3.130 HYPRE_SStructSplitCreate()

```
HYPRE_Int HYPRE_SStructSplitCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.131 HYPRE_SStructSplitDestroy()

```
HYPRE_Int HYPRE_SStructSplitDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.132 HYPRE_SStructSplitGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructSplitGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.5.3.133 HYPRE_SStructSplitGetNumIterations()

```
HYPRE_Int HYPRE_SStructSplitGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.5.3.134 HYPRE_SStructSplitSetMaxIter()

```
HYPRE_Int HYPRE_SStructSplitSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.5.3.135 HYPRE_SStructSplitSetNonZeroGuess()

```
HYPRE_Int HYPRE_SStructSplitSetNonZeroGuess (
            HYPRE_SStructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.5.3.136 HYPRE_SStructSplitSetStructSolver()

```
HYPRE_Int HYPRE_SStructSplitSetStructSolver (
            HYPRE_SStructSolver solver,
            HYPRE_Int ssolver )
```

(Optional) Set up the type of diagonal struct solver. Either {\tt ssolver} is set to {\tt HYPRE_SMG} or {\tt HYPRE_PFMG}.

2.5.3.137 HYPRE_SStructSplitSetTol()

```
HYPRE_Int HYPRE_SStructSplitSetTol (
            HYPRE_SStructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.5.3.138 HYPRE_SStructSplitSetup()

```
HYPRE_Int HYPRE_SStructSplitSetup (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.5.3.139 HYPRE_SStructSplitSetZeroGuess()

```
HYPRE_Int HYPRE_SStructSplitSetZeroGuess (
            HYPRE_SStructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.5.3.140 HYPRE_SStructSplitSolve()

```
HYPRE_Int HYPRE_SStructSplitSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Solve the system.

2.5.3.141 HYPRE_SStructSysPFMGCreate()

```
HYPRE_Int HYPRE_SStructSysPFMGCreate (
            MPI_Comm comm,
            HYPRE_SStructSolver * solver )
```

Create a solver object.

2.5.3.142 HYPRE_SStructSysPFMGDestroy()

```
HYPRE_Int HYPRE_SStructSysPFMGDestroy (
            HYPRE_SStructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.5.3.143 HYPRE_SStructSysPFMGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_SStructSysPFMGGetFinalRelativeResidualNorm (
            HYPRE_SStructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.5.3.144 HYPRE_SStructSysPFMGGetNumIterations()

```
HYPRE_Int HYPRE_SStructSysPFMGGetNumIterations (
            HYPRE_SStructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.5.3.145 HYPRE_SStructSysPFMGSetDxyz()

```
HYPRE_Int HYPRE_SStructSysPFMGSetDxyz (
            HYPRE_SStructSolver solver,
            HYPRE_Real * dxyz )
```

2.5.3.146 HYPRE_SStructSysPFMGSetJacobiWeight()

```
HYPRE_Int HYPRE_SStructSysPFMGSetJacobiWeight (
            HYPRE_SStructSolver solver,
            HYPRE_Real weight )
```

(Optional) Set Jacobi Weight.

2.5.3.147 HYPRE_SStructSysPFMGSetLogging()

```
HYPRE_Int HYPRE_SStructSysPFMGSetLogging (
            HYPRE_SStructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.5.3.148 HYPRE_SStructSysPFMGSetMaxIter()

```
HYPRE_Int HYPRE_SStructSysPFMGSetMaxIter (
            HYPRE_SStructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.5.3.149 HYPRE_SStructSysPFMGSetNonZeroGuess()

```
HYPRE_Int HYPRE_SStructSysPFMGSetNonZeroGuess (
            HYPRE_SStructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.5.3.150 HYPRE_SStructSysPFMGSetNumPostRelax()

```
HYPRE_Int HYPRE_SStructSysPFMGSetNumPostRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_post_relax )
```

(Optional) Set number of relaxation sweeps after coarse-grid correction.

2.5.3.151 HYPRE_SStructSysPFMGSetNumPreRelax()

```
HYPRE_Int HYPRE_SStructSysPFMGSetNumPreRelax (
            HYPRE_SStructSolver solver,
            HYPRE_Int num_pre_relax )
```

(Optional) Set number of relaxation sweeps before coarse-grid correction.

2.5.3.152 HYPRE_SStructSysPFMGSetPrintLevel()

```
HYPRE_Int HYPRE_SStructSysPFMGSetPrintLevel (
            HYPRE_SStructSolver solver,
            HYPRE_Int print_level )
```

(Optional) Set the amount of printing to do to the screen.

2.5.3.153 HYPRE_SStructSysPFMGSetRelaxType()

```
HYPRE_Int HYPRE_SStructSysPFMGSetRelaxType (
            HYPRE_SStructSolver solver,
            HYPRE_Int relax_type )
```

(Optional) Set relaxation type.

Current relaxation methods set by {\tt relax_type} are:

\begin{tabular}{l – }l} 0 & Jacobi \ 1 & Weighted Jacobi (default) \ 2 & Red/Black Gauss-Seidel (symmetric: RB pre-relaxation, BR post-relaxation) \ \end{tabular}

2.5.3.154 HYPRE_SStructSysPFMGSetRelChange()

```
HYPRE_Int HYPRE_SStructSysPFMGSetRelChange (
             HYPRE_SStructSolver solver,
             HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.5.3.155 HYPRE_SStructSysPFMGSetSkipRelax()

```
HYPRE_Int HYPRE_SStructSysPFMGSetSkipRelax (
             HYPRE_SStructSolver solver,
             HYPRE_Int skip_relax )
```

(Optional) Skip relaxation on certain grids for isotropic problems. This can greatly improve efficiency by eliminating unnecessary relaxations when the underlying problem is isotropic.

2.5.3.156 HYPRE_SStructSysPFMGSetTol()

```
HYPRE_Int HYPRE_SStructSysPFMGSetTol (
             HYPRE_SStructSolver solver,
             HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.5.3.157 HYPRE_SStructSysPFMGSetup()

```
HYPRE_Int HYPRE_SStructSysPFMGSetup (
             HYPRE_SStructSolver solver,
             HYPRE_SStructMatrix A,
             HYPRE_SStructVector b,
             HYPRE_SStructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.5.3.158 HYPRE_SStructSysPFMGSetZeroGuess()

```
HYPRE_Int HYPRE_SStructSysPFMGSetZeroGuess (
             HYPRE_SStructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.5.3.159 HYPRE_SStructSysPFMGSolve()

```
HYPRE_Int HYPRE_SStructSysPFMGSolve (
            HYPRE_SStructSolver solver,
            HYPRE_SStructMatrix A,
            HYPRE_SStructVector b,
            HYPRE_SStructVector x )
```

Solve the system.

2.5.3.160 hypre_SStructVectorSetRandomValues()

```
HYPRE_Int hypre_SStructVectorSetRandomValues (
            hypre_SStructVector * vector,
            HYPRE_Int seed )
```

2.6 HYPRE_sstruct_mv.h File Reference

SStruct Grids

- #define HYPRE_SSTRUCT_VARIABLE_UNDEFINED -1
- #define HYPRE_SSTRUCT_VARIABLE_CELL 0
- #define HYPRE_SSTRUCT_VARIABLE_NODE 1
- #define HYPRE_SSTRUCT_VARIABLE_XFACE 2
- #define HYPRE_SSTRUCT_VARIABLE_YFACE 3
- #define HYPRE_SSTRUCT_VARIABLE_ZFACE 4
- #define HYPRE_SSTRUCT_VARIABLE_XEDGE 5
- #define HYPRE_SSTRUCT_VARIABLE_YEDGE 6
- #define HYPRE_SSTRUCT_VARIABLE_ZEDGE 7
- typedef struct hypre_SStructGrid_struct ∗ HYPRE_SStructGrid
- typedef HYPRE_Int HYPRE_SStructVariable
- HYPRE_Int HYPRE_SStructGridCreate (MPI_Comm comm, HYPRE_Int ndim, HYPRE_Int nparts, HYPRE_SStructGrid ∗grid)
- HYPRE_Int HYPRE_SStructGridDestroy (HYPRE_SStructGrid grid)
- HYPRE_Int HYPRE_SStructGridSetExtents (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper)
- HYPRE_Int HYPRE_SStructGridSetVariables (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int nvars, HYPRE_SStructVariable ∗vartypes)
- HYPRE_Int HYPRE_SStructGridAddVariables (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Int nvars, HYPRE_SStructVariable ∗vartypes)
- HYPRE_Int HYPRE_SStructGridSetFEMOrdering (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗ordering)
- HYPRE_Int HYPRE_SStructGridSetNeighborPart (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int nbor_part, HYPRE_Int ∗nbor_ilower, HYPRE_Int ∗nbor_iupper, HYPRE_Int ∗index_map, HYPRE_Int ∗index_dir)
- HYPRE_Int HYPRE_SStructGridSetSharedPart (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int ∗offset, HYPRE_Int shared_part, HYPRE_Int ∗shared_ilower, HYPRE_Int ∗shared_iupper, HYPRE_Int ∗shared_offset, HYPRE_Int ∗index_map, HYPRE_Int ∗index_dir)
- HYPRE_Int HYPRE_SStructGridAddUnstructuredPart (HYPRE_SStructGrid grid, HYPRE_Int ilower, HYPRE_Int iupper)
- HYPRE_Int HYPRE_SStructGridAssemble (HYPRE_SStructGrid grid)
- HYPRE_Int HYPRE_SStructGridSetPeriodic (HYPRE_SStructGrid grid, HYPRE_Int part, HYPRE_Int ∗periodic)
- HYPRE_Int HYPRE_SStructGridSetNumGhost (HYPRE_SStructGrid grid, HYPRE_Int ∗num_ghost)

SStruct Stencils

- typedef struct hypre_SStructStencil_struct * HYPRE_SStructStencil
- HYPRE_Int HYPRE_SStructStencilCreate (HYPRE_Int ndim, HYPRE_Int size, HYPRE_SStructStencil *stencil)
- HYPRE_Int HYPRE_SStructStencilDestroy (HYPRE_SStructStencil stencil)
- HYPRE_Int HYPRE_SStructStencilSetEntry (HYPRE_SStructStencil stencil, HYPRE_Int entry, HYPRE_Int *offset, HYPRE_Int var)

SStruct Graphs

- typedef struct hypre_SStructGraph_struct * HYPRE_SStructGraph
- HYPRE_Int HYPRE_SStructGraphCreate (MPI_Comm comm, HYPRE_SStructGrid grid, HYPRE_SStructGraph *graph)
- HYPRE_Int HYPRE_SStructGraphDestroy (HYPRE_SStructGraph graph)
- HYPRE_Int HYPRE_SStructGraphSetDomainGrid (HYPRE_SStructGraph graph, HYPRE_SStructGrid domain_grid)
- HYPRE_Int HYPRE_SStructGraphSetStencil (HYPRE_SStructGraph graph, HYPRE_Int part, HYPRE_Int var, HYPRE_SStructStencil stencil)
- HYPRE_Int HYPRE_SStructGraphSetFEM (HYPRE_SStructGraph graph, HYPRE_Int part)
- HYPRE_Int HYPRE_SStructGraphSetFEMSparsity (HYPRE_SStructGraph graph, HYPRE_Int part, HYPRE_Int nsparse, HYPRE_Int *sparsity)
- HYPRE_Int HYPRE_SStructGraphAddEntries (HYPRE_SStructGraph graph, HYPRE_Int part, HYPRE_Int *index, HYPRE_Int var, HYPRE_Int to_part, HYPRE_Int *to_index, HYPRE_Int to_var)
- HYPRE_Int HYPRE_SStructGraphAssemble (HYPRE_SStructGraph graph)
- HYPRE_Int HYPRE_SStructGraphSetObjectType (HYPRE_SStructGraph graph, HYPRE_Int type)

SStruct Matrices

- typedef struct hypre_SStructMatrix_struct * HYPRE_SStructMatrix
- HYPRE_Int HYPRE_SStructMatrixCreate (MPI_Comm comm, HYPRE_SStructGraph graph, HYPRE_SStructMatrix *matrix)
- HYPRE_Int HYPRE_SStructMatrixDestroy (HYPRE_SStructMatrix matrix)
- HYPRE_Int HYPRE_SStructMatrixInitialize (HYPRE_SStructMatrix matrix)
- HYPRE_Int HYPRE_SStructMatrixSetValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *index, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixAddToValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *index, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixAddFEMValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *index, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixGetValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *index, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixGetFEMValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *index, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixSetBoxValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructMatrixAddToBoxValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)

- HYPRE_Int HYPRE_SStructMatrixSetBoxValues2 (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int ∗entries, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructMatrixAddToBoxValues2 (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int ∗entries, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructMatrixAssemble (HYPRE_SStructMatrix matrix)
- HYPRE_Int HYPRE_SStructMatrixGetBoxValues (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int ∗entries, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructMatrixGetBoxValues2 (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int nentries, HYPRE_Int ∗entries, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructMatrixSetSymmetric (HYPRE_SStructMatrix matrix, HYPRE_Int part, HYPRE_Int var, HYPRE_Int to_var, HYPRE_Int symmetric)
- HYPRE_Int HYPRE_SStructMatrixSetNSSymmetric (HYPRE_SStructMatrix matrix, HYPRE_Int symmetric)
- HYPRE_Int HYPRE_SStructMatrixSetObjectType (HYPRE_SStructMatrix matrix, HYPRE_Int type)
- HYPRE_Int HYPRE_SStructMatrixGetObject (HYPRE_SStructMatrix matrix, void ∗∗object)
- HYPRE_Int HYPRE_SStructMatrixPrint (const char ∗filename, HYPRE_SStructMatrix matrix, HYPRE_Int all)

SStruct Vectors

- typedef struct hypre_SStructVector_struct ∗ HYPRE_SStructVector
- HYPRE_Int HYPRE_SStructVectorCreate (MPI_Comm comm, HYPRE_SStructGrid grid, HYPRE_SStructVector ∗vector)
- HYPRE_Int HYPRE_SStructVectorDestroy (HYPRE_SStructVector vector)
- HYPRE_Int HYPRE_SStructVectorInitialize (HYPRE_SStructVector vector)
- HYPRE_Int HYPRE_SStructVectorSetValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Int var, HYPRE_Complex ∗value)
- HYPRE_Int HYPRE_SStructVectorAddToValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Int var, HYPRE_Complex ∗value)
- HYPRE_Int HYPRE_SStructVectorAddFEMValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorGetValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Int var, HYPRE_Complex ∗value)
- HYPRE_Int HYPRE_SStructVectorGetFEMValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗index, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorSetBoxValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorAddToBoxValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorSetBoxValues2 (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorAddToBoxValues2 (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_SStructVectorAssemble (HYPRE_SStructVector vector)
- HYPRE_Int HYPRE_SStructVectorGetBoxValues (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int var, HYPRE_Complex ∗values)

- HYPRE_Int HYPRE_SStructVectorGetBoxValues2 (HYPRE_SStructVector vector, HYPRE_Int part, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int var, HYPRE_Int *vilower, HYPRE_Int *viupper, HYPRE_Complex *values)
- HYPRE_Int HYPRE_SStructVectorGather (HYPRE_SStructVector vector)
- HYPRE_Int HYPRE_SStructVectorSetObjectType (HYPRE_SStructVector vector, HYPRE_Int type)
- HYPRE_Int HYPRE_SStructVectorGetObject (HYPRE_SStructVector vector, void **object)
- HYPRE_Int HYPRE_SStructVectorPrint (const char *filename, HYPRE_SStructVector vector, HYPRE_Int all)

2.6.1 Macro Definition Documentation

2.6.1.1 HYPRE_SSTRUCT_VARIABLE_CELL

```
#define HYPRE_SSTRUCT_VARIABLE_CELL 0
```

2.6.1.2 HYPRE_SSTRUCT_VARIABLE_NODE

```
#define HYPRE_SSTRUCT_VARIABLE_NODE 1
```

2.6.1.3 HYPRE_SSTRUCT_VARIABLE_UNDEFINED

```
#define HYPRE_SSTRUCT_VARIABLE_UNDEFINED -1
```

2.6.1.4 HYPRE_SSTRUCT_VARIABLE_XEDGE

```
#define HYPRE_SSTRUCT_VARIABLE_XEDGE 5
```

2.6.1.5 HYPRE_SSTRUCT_VARIABLE_XFACE

```
#define HYPRE_SSTRUCT_VARIABLE_XFACE 2
```

2.6.1.6 HYPRE_SSTRUCT_VARIABLE_YEDGE

```
#define HYPRE_SSTRUCT_VARIABLE_YEDGE 6
```

2.6.1.7 HYPRE_SSTRUCT_VARIABLE_YFACE

```
#define HYPRE_SSTRUCT_VARIABLE_YFACE 3
```

2.6.1.8 HYPRE_SSTRUCT_VARIABLE_ZEDGE

```
#define HYPRE_SSTRUCT_VARIABLE_ZEDGE 7
```

2.6.1.9 HYPRE_SSTRUCT_VARIABLE_ZFACE

```
#define HYPRE_SSTRUCT_VARIABLE_ZFACE 4
```

2.6.2 Typedef Documentation

2.6.2.1 HYPRE_SStructGraph

```
typedef struct hypre_SStructGraph_struct* HYPRE_SStructGraph
```

The graph object is used to describe the nonzero structure of a matrix.

2.6.2.2 HYPRE_SStructGrid

```
typedef struct hypre_SStructGrid_struct* HYPRE_SStructGrid
```

A grid object is constructed out of several structured `parts'' and an optional unstructured`part''. Each structured part has its own abstract index space.

2.6.2.3 HYPRE_SStructMatrix

```
typedef struct hypre_SStructMatrix_struct* HYPRE_SStructMatrix
```

The matrix object.

2.6.2.4 HYPRE_SStructStencil

```
typedef struct hypre_SStructStencil_struct* HYPRE_SStructStencil
```

The stencil object.

2.6.2.5 HYPRE_SStructVariable

```
typedef HYPRE_Int HYPRE_SStructVariable
```

An enumerated type that supports cell centered, node centered, face centered, and edge centered variables. Face centered variables are split into x-face, y-face, and z-face variables, and edge centered variables are split into x-edge, y-edge, and z-edge variables. The edge centered variable types are only used in 3D. In 2D, edge centered variables are handled by the face centered types.

Variables are referenced relative to an abstract (cell centered) index in the following way: \begin{itemize} \item cell centered variables are aligned with the index; \item node centered variables are aligned with the cell corner at relative index (1/2, 1/2, 1/2); \item x-face, y-face, and z-face centered variables are aligned with the faces at relative indexes (1/2, 0, 0), (0, 1/2, 0), and (0, 0, 1/2), respectively; \item x-edge, y-edge, and z-edge centered variables are aligned with the edges at relative indexes (0, 1/2, 1/2), (1/2, 0, 1/2), and (1/2, 1/2, 0), respectively. \end{itemize}

The supported identifiers are: \begin{itemize} \item {\tt HYPRE_SSTRUCT_VARIABLE_CELL} \item {\tt HYPRE↩_SSTRUCT_VARIABLE_NODE} \item {\tt HYPRE_SSTRUCT_VARIABLE_XFACE} \item {\tt HYPRE_SSTRUCT↩_VARIABLE_YFACE} \item {\tt HYPRE_SSTRUCT_VARIABLE_ZFACE} \item {\tt HYPRE_SSTRUCT_VARIABLE↩_XEDGE} \item {\tt HYPRE_SSTRUCT_VARIABLE_YEDGE} \item {\tt HYPRE_SSTRUCT_VARIABLE_ZEDGE} \end{itemize}

NOTE: Although variables are referenced relative to a unique abstract cell-centered index, some variables are associated with multiple grid cells. For example, node centered variables in 3D are associated with 8 cells (away from boundaries). Although grid cells are distributed uniquely to different processes, variables may be owned by multiple processes because they may be associated with multiple cells.

2.6.2.6 HYPRE_SStructVector

```
typedef struct hypre_SStructVector_struct* HYPRE_SStructVector
```

The vector object.

2.6.3 Function Documentation

2.6.3.1 HYPRE_SStructGraphAddEntries()

```
HYPRE_Int HYPRE_SStructGraphAddEntries (
            HYPRE_SStructGraph graph,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Int to_part,
            HYPRE_Int * to_index,
            HYPRE_Int to_var )
```

Add a non-stencil graph entry at a particular index. This graph entry is appended to the existing graph entries, and is referenced as such.

NOTE: Users are required to set graph entries on all processes that own the associated variables. This means that some data will be multiply defined.

2.6.3.2 HYPRE_SStructGraphAssemble()

```
HYPRE_Int HYPRE_SStructGraphAssemble (
            HYPRE_SStructGraph graph )
```

Finalize the construction of the graph before using.

2.6.3.3 HYPRE_SStructGraphCreate()

```
HYPRE_Int HYPRE_SStructGraphCreate (
            MPI_Comm comm,
            HYPRE_SStructGrid grid,
            HYPRE_SStructGraph * graph )
```

Create a graph object.

2.6.3.4 HYPRE_SStructGraphDestroy()

```
HYPRE_Int HYPRE_SStructGraphDestroy (
            HYPRE_SStructGraph graph )
```

Destroy a graph object.

2.6.3.5 HYPRE_SStructGraphSetDomainGrid()

```
HYPRE_Int HYPRE_SStructGraphSetDomainGrid (
            HYPRE_SStructGraph graph,
            HYPRE_SStructGrid domain_grid )
```

Set the domain grid.

2.6.3.6 HYPRE_SStructGraphSetFEM()

```
HYPRE_Int HYPRE_SStructGraphSetFEM (
            HYPRE_SStructGraph graph,
            HYPRE_Int part )
```

Indicate that an FEM approach will be used to set matrix values on this part.

2.6.3.7 HYPRE_SStructGraphSetFEMSparsity()

```
HYPRE_Int HYPRE_SStructGraphSetFEMSparsity (
            HYPRE_SStructGraph graph,
            HYPRE_Int part,
            HYPRE_Int nsparse,
            HYPRE_Int * sparsity )
```

Set the finite element stiffness matrix sparsity. This overrides the default full sparsity pattern described below.

Array {\tt sparsity} contains {\tt nsparse} row/column tuples (I,J) that indicate the nonzeroes of the local stiffness matrix. The layout of the values passed into the routine \Ref{HYPRE_SStructMatrixAddFEMValues} is determined here.

The default sparsity is full (each variable is coupled to all others), and the values passed into the routine \Ref{HYPRE↩ _SStructMatrixAddFEMValues} are assumed to be by rows (that is, column indices vary fastest).

2.6.3.8 HYPRE_SStructGraphSetObjectType()

```
HYPRE_Int HYPRE_SStructGraphSetObjectType (
            HYPRE_SStructGraph graph,
            HYPRE_Int type )
```

Set the storage type of the associated matrix object. It is used before AddEntries and Assemble to compute the right ranks in the graph.

NOTE: This routine is only necessary for implementation reasons, and will eventually be removed.

See also

HYPRE_SStructMatrixSetObjectType

2.6.3.9 HYPRE_SStructGraphSetStencil()

```
HYPRE_Int HYPRE_SStructGraphSetStencil (
            HYPRE_SStructGraph graph,
            HYPRE_Int part,
            HYPRE_Int var,
            HYPRE_SStructStencil stencil )
```

Set the stencil for a variable on a structured part of the grid.

2.6.3.10 HYPRE_SStructGridAddUnstructuredPart()

```
HYPRE_Int HYPRE_SStructGridAddUnstructuredPart (
            HYPRE_SStructGrid grid,
            HYPRE_Int ilower,
            HYPRE_Int iupper )
```

Add an unstructured part to the grid. The variables in the unstructured part of the grid are referenced by a global rank between 0 and the total number of unstructured variables minus one. Each process owns some unique consecutive range of variables, defined by {\tt ilower} and {\tt iupper}.

NOTE: This is just a placeholder. This part of the interface is not finished.

2.6.3.11 HYPRE_SStructGridAddVariables()

```
HYPRE_Int HYPRE_SStructGridAddVariables (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int nvars,
            HYPRE_SStructVariable * vartypes )
```

Describe additional variables that live at a particular index. These variables are appended to the array of variables set in \Ref{HYPRE_SStructGridSetVariables}, and are referenced as such.

NOTE: This routine is not yet supported.

2.6.3.12 HYPRE_SStructGridAssemble()

```
HYPRE_Int HYPRE_SStructGridAssemble (
            HYPRE_SStructGrid grid )
```

Finalize the construction of the grid before using.

2.6.3.13 HYPRE_SStructGridCreate()

```
HYPRE_Int HYPRE_SStructGridCreate (
            MPI_Comm comm,
            HYPRE_Int ndim,
            HYPRE_Int nparts,
            HYPRE_SStructGrid * grid )
```

Create an {\tt ndim}-dimensional grid object with {\tt nparts} structured parts.

2.6.3.14 HYPRE_SStructGridDestroy()

```
HYPRE_Int HYPRE_SStructGridDestroy (
            HYPRE_SStructGrid grid )
```

Destroy a grid object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.6.3.15 HYPRE_SStructGridSetExtents()

```
HYPRE_Int HYPRE_SStructGridSetExtents (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper )
```

Set the extents for a box on a structured part of the grid.

2.6.3.16 HYPRE_SStructGridSetFEMOrdering()

```
HYPRE_Int HYPRE_SStructGridSetFEMOrdering (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * ordering )
```

Set the ordering of variables in a finite element problem. This overrides the default ordering described below.

Array {\tt ordering} is composed of blocks of size (1 + {\tt ndim}). Each block indicates a specific variable in the element and the ordering of the blocks defines the ordering of the variables. A block contains a variable number followed by an offset direction relative to the element's center. For example, a block containing (2, 1, -1, 0) means variable 2 on the edge located in the (1, -1, 0) direction from the center of the element. Note that here variable 2 must be of type {\tt ZEDGE} for this to make sense. The {\tt ordering} array must account for all variables in the element. This routine can only be called after \Ref{HYPRE_SStructGridSetVariables}.

The default ordering for element variables (var, i, j, k) varies fastest in index i, followed by j, then k, then var. For example, if var 0, var 1, and var 2 are declared to be XFACE, YFACE, and NODE variables, respectively, then the default ordering (in 2D) would first list the two XFACE variables, then the two YFACE variables, then the four NODE variables as follows:

(0,-1,0), (0,1,0), (1,0,-1), (1,0,1), (2,-1,-1), (2,1,-1), (2,-1,1), (2,1,1)

2.6.3.17 HYPRE_SStructGridSetNeighborPart()

```
HYPRE_Int HYPRE_SStructGridSetNeighborPart (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nbor_part,
            HYPRE_Int * nbor_ilower,
            HYPRE_Int * nbor_iupper,
            HYPRE_Int * index_map,
            HYPRE_Int * index_dir )
```

Describe how regions just outside of a part relate to other parts. This is done a box at a time.

Parts {\tt part} and {\tt nbor_part} must be different, except in the case where only cell-centered data is used.

Indexes should increase from {\tt ilower} to {\tt iupper}. It is not necessary that indexes increase from {\tt nbor_ilower} to {\tt nbor_iupper}.

The {\tt index_map} describes the mapping of indexes 0, 1, and 2 on part {\tt part} to the corresponding indexes on part {\tt nbor_part}. For example, triple (1, 2, 0) means that indexes 0, 1, and 2 on part {\tt part} map to indexes 1, 2, and 0 on part {\tt nbor_part}, respectively.

The {\tt index_dir} describes the direction of the mapping in {\tt index_map}. For example, triple (1, 1, -1) means that for indexes 0 and 1, increasing values map to increasing values on {\tt nbor_part}, while for index 2, decreasing values map to increasing values.

NOTE: All parts related to each other via this routine must have an identical list of variables and variable types. For example, if part 0 has only two variables on it, a cell centered variable and a node centered variable, and we declare part 1 to be a neighbor of part 0, then part 1 must also have only two variables on it, and they must be of type cell and node. In addition, variables associated with FACEs or EDGEs must be grouped together and listed in X, Y, Z order. This is to enable the code to correctly associate variables on one part with variables on its neighbor part when a coordinate transformation is specified. For example, an XFACE variable on one part may correspond to a YFACE variable on a neighbor part under a particular tranformation, and the code determines this association by assuming that the variable lists are as noted here.

2.6.3.18 HYPRE_SStructGridSetNumGhost()

```
HYPRE_Int HYPRE_SStructGridSetNumGhost (
            HYPRE_SStructGrid grid,
            HYPRE_Int * num_ghost )
```

Setting ghost in the sgrids.

2.6.3.19 HYPRE_SStructGridSetPeriodic()

```
HYPRE_Int HYPRE_SStructGridSetPeriodic (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * periodic )
```

Set the periodicity on a particular part.

The argument {\tt periodic} is an {\tt ndim}-dimensional integer array that contains the periodicity for each dimension. A zero value for a dimension means non-periodic, while a nonzero value means periodic and contains the actual period. For example, periodicity in the first and third dimensions for a 10x11x12 part is indicated by the array [10,0,12].

NOTE: Some of the solvers in hypre have power-of-two restrictions on the size of the periodic dimensions.

2.6.3.20 HYPRE_SStructGridSetSharedPart()

```
HYPRE_Int HYPRE_SStructGridSetSharedPart (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int * offset,
            HYPRE_Int shared_part,
            HYPRE_Int * shared_ilower,
            HYPRE_Int * shared_iupper,
            HYPRE_Int * shared_offset,
            HYPRE_Int * index_map,
            HYPRE_Int * index_dir )
```

Describe how regions inside a part are shared with regions in other parts.

Parts {\tt part} and {\tt shared_part} must be different.

Indexes should increase from {\tt ilower} to {\tt iupper}. It is not necessary that indexes increase from {\tt shared_↩ ilower} to {\tt shared_iupper}. This is to maintain consistency with the {\tt SetNeighborPart} function, which is also able to describe shared regions but in a more limited fashion.

The {\tt offset} is a triple (in 3D) used to indicate the dimensionality of the shared set of data and its position with respect to the box extents {\tt ilower} and {\tt iupper} on part {\tt part}. The dimensionality is given by the number of 0's in the triple, and the position is given by plus or minus 1's. For example: (0, 0, 0) indicates sharing of all data in the given box; (1, 0, 0) indicates sharing of data on the faces in the (1, 0, 0) direction; (1, 0, -1) indicates sharing of data on the edges in the (1, 0, -1) direction; and (1, -1, 1) indicates sharing of data on the nodes in the (1, -1, 1) direction. To ensure the dimensionality, it is required that for every nonzero entry, the corresponding extents of the box are the same. For example, if {\tt offset} is (0, 1, 0), then (2, 1, 3) and (10, 1, 15) are valid box extents, whereas (2, 1, 3) and (10, 7, 15) are invalid (because 1 and 7 are not the same).

The {\tt shared_offset} is used in the same way as {\tt offset}, but with respect to the box extents {\tt shared_ilower} and {\tt shared_iupper} on part {\tt shared_part}.

The {\tt index_map} describes the mapping of indexes 0, 1, and 2 on part {\tt part} to the corresponding indexes on part {\tt shared_part}. For example, triple (1, 2, 0) means that indexes 0, 1, and 2 on part {\tt part} map to indexes 1, 2, and 0 on part {\tt shared_part}, respectively.

The {\tt index_dir} describes the direction of the mapping in {\tt index_map}. For example, triple (1, 1, -1) means that for indexes 0 and 1, increasing values map to increasing values on {\tt shared_part}, while for index 2, decreasing values map to increasing values.

NOTE: All parts related to each other via this routine must have an identical list of variables and variable types. For example, if part 0 has only two variables on it, a cell centered variable and a node centered variable, and we declare part 1 to have shared regions with part 0, then part 1 must also have only two variables on it, and they must be of type cell and node. In addition, variables associated with FACEs or EDGEs must be grouped together and listed in X, Y, Z order. This is to enable the code to correctly associate variables on one part with variables on a shared part when a coordinate transformation is specified. For example, an XFACE variable on one part may correspond to a YFACE variable on a shared part under a particular tranformation, and the code determines this association by assuming that the variable lists are as noted here.

2.6.3.21 HYPRE_SStructGridSetVariables()

```
HYPRE_Int HYPRE_SStructGridSetVariables (
            HYPRE_SStructGrid grid,
            HYPRE_Int part,
            HYPRE_Int nvars,
            HYPRE_SStructVariable * vartypes )
```

Describe the variables that live on a structured part of the grid.

2.6.3.22 HYPRE_SStructMatrixAddFEMValues()

```
HYPRE_Int HYPRE_SStructMatrixAddFEMValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Complex * values )
```

Add finite element stiffness matrix coefficients index by index. The layout of the data in {\tt values} is determined by the routines \Ref{HYPRE_SStructGridSetFEMOrdering} and \Ref{HYPRE_SStructGraphSetFEMSparsity}.

2.6.3.23 HYPRE_SStructMatrixAddToBoxValues()

```
HYPRE_Int HYPRE_SStructMatrixAddToBoxValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Add to matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructMatrixSetBox↩ Values}.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of stencil type. Also, they must all represent couplings to the same variable type.

2.6.3.24 HYPRE_SStructMatrixAddToBoxValues2()

```
HYPRE_Int HYPRE_SStructMatrixAddToBoxValues2 (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Add to matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructMatrixSetBox↩ Values2}.

2.6.3.25 HYPRE_SStructMatrixAddToValues()

```
HYPRE_Int HYPRE_SStructMatrixAddToValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Add to matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_SStructMatrixAddToBoxValues} to set coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type.

2.6.3.26 HYPRE_SStructMatrixAssemble()

```
HYPRE_Int HYPRE_SStructMatrixAssemble (
            HYPRE_SStructMatrix matrix )
```

Finalize the construction of the matrix before using.

2.6.3.27 HYPRE_SStructMatrixCreate()

```
HYPRE_Int HYPRE_SStructMatrixCreate (
            MPI_Comm comm,
            HYPRE_SStructGraph graph,
            HYPRE_SStructMatrix * matrix )
```

Create a matrix object.

2.6.3.28 HYPRE_SStructMatrixDestroy()

```
HYPRE_Int HYPRE_SStructMatrixDestroy (
            HYPRE_SStructMatrix matrix )
```

Destroy a matrix object.

2.6.3.29 HYPRE_SStructMatrixGetBoxValues()

```
HYPRE_Int HYPRE_SStructMatrixGetBoxValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Get matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructMatrixSetBox←Values}.

NOTE: Users may get values on any process that owns the associated variables.

NOTE: The entries in this routine must all be of stencil type. Also, they must all represent couplings to the same variable type.

2.6.3.30 HYPRE_SStructMatrixGetBoxValues2()

```
HYPRE_Int HYPRE_SStructMatrixGetBoxValues2 (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Get matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructMatrixSetBox←Values2}.

2.6.3.31 HYPRE_SStructMatrixGetFEMValues()

```
HYPRE_Int HYPRE_SStructMatrixGetFEMValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Complex * values )
```

Get finite element stiffness matrix coefficients index by index. The layout of the data in {\tt values} is determined by the routines \Ref{HYPRE_SStructGridSetFEMOrdering} and \Ref{HYPRE_SStructGraphSetFEMSparsity}.

2.6.3.32 HYPRE_SStructMatrixGetObject()

```
HYPRE_Int HYPRE_SStructMatrixGetObject (
            HYPRE_SStructMatrix matrix,
            void ** object )
```

Get a reference to the constructed matrix object.

See also

HYPRE_SStructMatrixSetObjectType

2.6.3.33 HYPRE_SStructMatrixGetValues()

```
HYPRE_Int HYPRE_SStructMatrixGetValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Get matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_SStructMatrixGetBoxValues} to get coefficients a box at a time.

NOTE: Users may get values on any process that owns the associated variables.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

2.6.3.34 HYPRE_SStructMatrixInitialize()

```
HYPRE_Int HYPRE_SStructMatrixInitialize (
            HYPRE_SStructMatrix matrix )
```

Prepare a matrix object for setting coefficient values.

2.6.3.35 HYPRE_SStructMatrixPrint()

```
HYPRE_Int HYPRE_SStructMatrixPrint (
            const char * filename,
            HYPRE_SStructMatrix matrix,
            HYPRE_Int all )
```

Print the matrix to file. This is mainly for debugging purposes.

2.6.3.36 HYPRE_SStructMatrixSetBoxValues()

```
HYPRE_Int HYPRE_SStructMatrixSetBoxValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Set matrix coefficients a box at a time. The data in {\tt values} is ordered as follows:

\begin{verbatim} m = 0; for (k = ilower[2]; k <= iupper[2]; k++) for (j = ilower[1]; j <= iupper[1]; j++) for (i = ilower[0]; i <= iupper[0]; i++) for (entry = 0; entry < nentries; entry++) { values[m] = ...; m++; } \end{verbatim}

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

2.6.3.37 HYPRE_SStructMatrixSetBoxValues2()

```
HYPRE_Int HYPRE_SStructMatrixSetBoxValues2 (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Set matrix coefficients a box at a time. The {\tt values} array is logically box shaped with value-box extents {\tt vilower} and {\tt viupper} that must contain the set-box extents {\tt ilower} and {\tt iupper} . The data in the {\tt values} array is ordered as in \Ref{HYPRE_SStructMatrixSetBoxValues}, but based on the value-box extents.

2.6.3.38 HYPRE_SStructMatrixSetNSSymmetric()

```
HYPRE_Int HYPRE_SStructMatrixSetNSSymmetric (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int symmetric )
```

Define symmetry properties for all non-stencil matrix entries.

2.6.3.39 HYPRE_SStructMatrixSetObjectType()

```
HYPRE_Int HYPRE_SStructMatrixSetObjectType (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int type )
```

Set the storage type of the matrix object to be constructed. Currently, {\tt type} can be either {\tt HYPRE_SSTRUCT} (the default), {\tt HYPRE_STRUCT}, or {\tt HYPRE_PARCSR}.

See also

HYPRE_SStructMatrixGetObject

2.6.3.40 HYPRE_SStructMatrixSetSymmetric()

```
HYPRE_Int HYPRE_SStructMatrixSetSymmetric (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int var,
            HYPRE_Int to_var,
            HYPRE_Int symmetric )
```

Define symmetry properties for the stencil entries in the matrix. The boolean argument {\tt symmetric} is applied to stencil entries on part {\tt part} that couple variable {\tt var} to variable {\tt to_var}. A value of -1 may be used for {\tt part}, {\tt var}, or {\tt to_var} to specify `'all''. For example, if {\tt part} and {\tt to_var} are set to -1, then the boolean is applied to stencil entries on all parts that couple variable {\tt var} to all other variables.

By default, matrices are assumed to be nonsymmetric. Significant storage savings can be made if the matrix is symmetric.

2.6.3.41 HYPRE_SStructMatrixSetValues()

```
HYPRE_Int HYPRE_SStructMatrixSetValues (
            HYPRE_SStructMatrix matrix,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Set matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_SStructMatrixSetBoxValues} to set coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

2.6.3.42 HYPRE_SStructStencilCreate()

```
HYPRE_Int HYPRE_SStructStencilCreate (
            HYPRE_Int ndim,
            HYPRE_Int size,
            HYPRE_SStructStencil * stencil )
```

Create a stencil object for the specified number of spatial dimensions and stencil entries.

2.6.3.43 HYPRE_SStructStencilDestroy()

```
HYPRE_Int HYPRE_SStructStencilDestroy (
            HYPRE_SStructStencil stencil )
```

Destroy a stencil object.

2.6.3.44 HYPRE_SStructStencilSetEntry()

```
HYPRE_Int HYPRE_SStructStencilSetEntry (
            HYPRE_SStructStencil stencil,
            HYPRE_Int entry,
            HYPRE_Int * offset,
            HYPRE_Int var )
```

Set a stencil entry.

2.6.3.45 HYPRE_SStructVectorAddFEMValues()

```
HYPRE_Int HYPRE_SStructVectorAddFEMValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Complex * values )
```

Add finite element vector coefficients index by index. The layout of the data in {\tt values} is determined by the routine \Ref{HYPRE_SStructGridSetFEMOrdering}.

2.6.3.46 HYPRE_SStructVectorAddToBoxValues()

```
HYPRE_Int HYPRE_SStructVectorAddToBoxValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Complex * values )
```

Add to vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructVectorSetBox↩ Values}.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

2.6.3.47 HYPRE_SStructVectorAddToBoxValues2()

```
HYPRE_Int HYPRE_SStructVectorAddToBoxValues2 (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Add to vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructVectorSetBox↩ Values2}.

2.6.3.48 HYPRE_SStructVectorAddToValues()

```
HYPRE_Int HYPRE_SStructVectorAddToValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Complex * value )
```

Add to vector coefficients index by index.

NOTE: For better efficiency, use \Ref{HYPRE_SStructVectorAddToBoxValues} to set coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

2.6.3.49 HYPRE_SStructVectorAssemble()

```
HYPRE_Int HYPRE_SStructVectorAssemble (
            HYPRE_SStructVector vector )
```

Finalize the construction of the vector before using.

2.6.3.50 HYPRE_SStructVectorCreate()

```
HYPRE_Int HYPRE_SStructVectorCreate (
            MPI_Comm comm,
            HYPRE_SStructGrid grid,
            HYPRE_SStructVector * vector )
```

Create a vector object.

2.6.3.51 HYPRE_SStructVectorDestroy()

```
HYPRE_Int HYPRE_SStructVectorDestroy (
            HYPRE_SStructVector vector )
```

Destroy a vector object.

2.6.3.52 HYPRE_SStructVectorGather()

```
HYPRE_Int HYPRE_SStructVectorGather (
            HYPRE_SStructVector vector )
```

Gather vector data so that efficient {\tt GetValues} can be done. This routine must be called prior to calling {\tt Get↩ Values} to ensure that correct and consistent values are returned, especially for non cell-centered data that is shared between more than one processor.

2.6.3.53 HYPRE_SStructVectorGetBoxValues()

```
HYPRE_Int HYPRE_SStructVectorGetBoxValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Complex * values )
```

Get vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructVectorSetBox↩ Values}. Users must first call the routine \Ref{HYPRE_SStructVectorGather} to ensure that data owned by multiple processes is correct.

NOTE: Users may only get values on processes that own the associated variables.

2.6.3.54 HYPRE_SStructVectorGetBoxValues2()

```
HYPRE_Int HYPRE_SStructVectorGetBoxValues2 (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Get vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_SStructVectorSetBox↩ Values2}.

2.6.3.55 HYPRE_SStructVectorGetFEMValues()

```
HYPRE_Int HYPRE_SStructVectorGetFEMValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Complex * values )
```

Get finite element vector coefficients index by index. The layout of the data in {\tt values} is determined by the routine \Ref{HYPRE_SStructGridSetFEMOrdering}. Users must first call the routine \Ref{HYPRE_SStructVectorGather} to ensure that data owned by multiple processes is correct.

2.6.3.56 HYPRE_SStructVectorGetObject()

```
HYPRE_Int HYPRE_SStructVectorGetObject (
            HYPRE_SStructVector vector,
            void ** object )
```

Get a reference to the constructed vector object.

See also

HYPRE_SStructVectorSetObjectType

2.6.3.57 HYPRE_SStructVectorGetValues()

```
HYPRE_Int HYPRE_SStructVectorGetValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Complex * value )
```

Get vector coefficients index by index. Users must first call the routine \Ref{HYPRE_SStructVectorGather} to ensure that data owned by multiple processes is correct.

NOTE: For better efficiency, use \Ref{HYPRE_SStructVectorGetBoxValues} to get coefficients a box at a time.

NOTE: Users may only get values on processes that own the associated variables.

2.6.3.58 HYPRE_SStructVectorInitialize()

```
HYPRE_Int HYPRE_SStructVectorInitialize (
            HYPRE_SStructVector vector )
```

Prepare a vector object for setting coefficient values.

2.6.3.59 HYPRE_SStructVectorPrint()

```
HYPRE_Int HYPRE_SStructVectorPrint (
            const char * filename,
            HYPRE_SStructVector vector,
            HYPRE_Int all )
```

Print the vector to file. This is mainly for debugging purposes.

2.6.3.60 HYPRE_SStructVectorSetBoxValues()

```
HYPRE_Int HYPRE_SStructVectorSetBoxValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Complex * values )
```

Set vector coefficients a box at a time. The data in {\tt values} is ordered as follows:

\begin{verbatim} m = 0; for (k = ilower[2]; k <= iupper[2]; k++) for (j = ilower[1]; j <= iupper[1]; j++) for (i = ilower[0]; i <= iupper[0]; i++) { values[m] = ...; m++; } \end{verbatim}

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

2.6.3.61 HYPRE_SStructVectorSetBoxValues2()

```
HYPRE_Int HYPRE_SStructVectorSetBoxValues2 (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int var,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Set vector coefficients a box at a time. The {\tt values} array is logically box shaped with value-box extents {\tt vilower} and {\tt viupper} that must contain the set-box extents {\tt ilower} and {\tt iupper} . The data in the {\tt values} array is ordered as in \Ref{HYPRE_SStructVectorSetBoxValues}, but based on the value-box extents.

2.6.3.62 HYPRE_SStructVectorSetObjectType()

```
HYPRE_Int HYPRE_SStructVectorSetObjectType (
            HYPRE_SStructVector vector,
            HYPRE_Int type )
```

Set the storage type of the vector object to be constructed. Currently, {\tt type} can be either {\tt HYPRE_SSTRUCT} (the default), {\tt HYPRE_STRUCT}, or {\tt HYPRE_PARCSR}.

See also

HYPRE_SStructVectorGetObject

2.6.3.63 HYPRE_SStructVectorSetValues()

```
HYPRE_Int HYPRE_SStructVectorSetValues (
            HYPRE_SStructVector vector,
            HYPRE_Int part,
            HYPRE_Int * index,
            HYPRE_Int var,
            HYPRE_Complex * value )
```

Set vector coefficients index by index.

NOTE: For better efficiency, use \Ref{HYPRE_SStructVectorSetBoxValues} to set coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

2.7 HYPRE_struct_ls.h File Reference

Functions

Struct Jacobi Solver

- HYPRE_Int HYPRE_StructJacobiCreate (MPI_Comm comm, HYPRE_StructSolver *solver)
- HYPRE_Int HYPRE_StructJacobiDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructJacobiSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructJacobiSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructJacobiSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructJacobiSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructJacobiSetZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructJacobiSetNonZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructJacobiGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int *num_↩ iterations)
- HYPRE_Int HYPRE_StructJacobiGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real *norm)

Struct PFMG Solver

PFMG is a semicoarsening multigrid solver that uses pointwise relaxation. For periodic problems, users should try to set the grid size in periodic dimensions to be as close to a power-of-two as possible. That is, if the grid size in a periodic dimension is given by $N = 2^m * M$ where M is not a power-of-two, then M should be as small as possible. Large values of M will generally result in slower convergence rates.

- HYPRE_Int HYPRE_StructPFMGCreate (MPI_Comm comm, HYPRE_StructSolver *solver)
- HYPRE_Int HYPRE_StructPFMGDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructPFMGSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructPFMGSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructPFMGSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructPFMGSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructPFMGSetMaxLevels (HYPRE_StructSolver solver, HYPRE_Int max_levels)

- HYPRE_Int HYPRE_StructPFMGSetRelChange (HYPRE_StructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_StructPFMGSetZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructPFMGSetNonZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructPFMGSetRelaxType (HYPRE_StructSolver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_StructPFMGSetJacobiWeight (HYPRE_StructSolver solver, HYPRE_Real weight)
- HYPRE_Int HYPRE_StructPFMGGetJacobiWeight (HYPRE_StructSolver solver, HYPRE_Real ∗weight)
- HYPRE_Int HYPRE_StructPFMGSetRAPType (HYPRE_StructSolver solver, HYPRE_Int rap_type)
- HYPRE_Int HYPRE_StructPFMGSetNumPreRelax (HYPRE_StructSolver solver, HYPRE_Int num_pre_relax)
- HYPRE_Int HYPRE_StructPFMGSetNumPostRelax (HYPRE_StructSolver solver, HYPRE_Int num_post_relax)
- HYPRE_Int HYPRE_StructPFMGSetSkipRelax (HYPRE_StructSolver solver, HYPRE_Int skip_relax)
- HYPRE_Int HYPRE_StructPFMGSetDxyz (HYPRE_StructSolver solver, HYPRE_Real ∗dxyz)
- HYPRE_Int HYPRE_StructPFMGSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructPFMGSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_StructPFMGGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_StructPFMGGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)

Struct SMG Solver

SMG is a semicoarsening multigrid solver that uses plane smoothing (in 3D). The plane smoother calls a 2D SMG algorithm with line smoothing, and the line smoother is cyclic reduction (1D SMG). For periodic problems, the grid size in periodic dimensions currently must be a power-of-two.

- HYPRE_Int HYPRE_StructSMGCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructSMGDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSMGSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructSMGSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructSMGSetMemoryUse (HYPRE_StructSolver solver, HYPRE_Int memory_use)
- HYPRE_Int HYPRE_StructSMGSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructSMGSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructSMGSetRelChange (HYPRE_StructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_StructSMGSetZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSMGSetNonZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSMGSetNumPreRelax (HYPRE_StructSolver solver, HYPRE_Int num_pre_relax)
- HYPRE_Int HYPRE_StructSMGSetNumPostRelax (HYPRE_StructSolver solver, HYPRE_Int num_post_relax)
- HYPRE_Int HYPRE_StructSMGSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructSMGSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_StructSMGGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_StructSMGGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)

Struct CycRed Solver

CycRed is a cyclic reduction solver that simultaneously solves a collection of 1D tridiagonal systems embedded in a d-dimensional grid.

- HYPRE_Int HYPRE_StructCycRedCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructCycRedDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructCycRedSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructCycRedSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructCycRedSetTDim (HYPRE_StructSolver solver, HYPRE_Int tdim)

- HYPRE_Int HYPRE_StructCycRedSetBase (HYPRE_StructSolver solver, HYPRE_Int ndim, HYPRE_Int ∗base_index, HYPRE_Int ∗base_stride)

Struct PCG Solver

These routines should be used in conjunction with the generic interface in \Ref{PCG Solver}.

- HYPRE_Int HYPRE_StructPCGCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructPCGDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructPCGSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructPCGSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructPCGSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructPCGSetAbsoluteTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructPCGSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructPCGSetTwoNorm (HYPRE_StructSolver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_StructPCGSetRelChange (HYPRE_StructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_StructPCGSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructPCGSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructPCGSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_StructPCGGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_StructPCGGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_StructPCGGetResidual (HYPRE_StructSolver solver, void ∗∗residual)
- HYPRE_Int HYPRE_StructDiagScaleSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector y, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructDiagScale (HYPRE_StructSolver solver, HYPRE_StructMatrix HA, HYPRE_StructVector Hy, HYPRE_StructVector Hx)

Struct GMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{GMRES Solver}.

- HYPRE_Int HYPRE_StructGMRESCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructGMRESDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructGMRESSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructGMRESSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructGMRESSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructGMRESSetAbsoluteTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructGMRESSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructGMRESSetKDim (HYPRE_StructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_StructGMRESSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructGMRESSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructGMRESSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_StructGMRESGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_iterations)
- HYPRE_Int HYPRE_StructGMRESGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_StructGMRESGetResidual (HYPRE_StructSolver solver, void ∗∗residual)

Struct FlexGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{FlexGMRES Solver}.

- HYPRE_Int HYPRE_StructFlexGMRESCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructFlexGMRESDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructFlexGMRESSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructFlexGMRESSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructFlexGMRESSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructFlexGMRESSetAbsoluteTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructFlexGMRESSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructFlexGMRESSetKDim (HYPRE_StructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_StructFlexGMRESSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructFlexGMRESSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructFlexGMRESSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_StructFlexGMRESGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_←iterations)
- HYPRE_Int HYPRE_StructFlexGMRESGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_StructFlexGMRESGetResidual (HYPRE_StructSolver solver, void ∗∗residual)
- HYPRE_Int HYPRE_StructFlexGMRESSetModifyPC (HYPRE_StructSolver solver, HYPRE_PtrToModifyPCFcn modify_pc)

Struct LGMRES Solver

These routines should be used in conjunction with the generic interface in \Ref{LGMRES Solver}.

- HYPRE_Int HYPRE_StructLGMRESCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructLGMRESDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructLGMRESSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructLGMRESSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructLGMRESSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructLGMRESSetAbsoluteTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructLGMRESSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructLGMRESSetKDim (HYPRE_StructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_StructLGMRESSetAugDim (HYPRE_StructSolver solver, HYPRE_Int aug_dim)
- HYPRE_Int HYPRE_StructLGMRESSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructLGMRESSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructLGMRESSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_StructLGMRESGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_←iterations)
- HYPRE_Int HYPRE_StructLGMRESGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_←Real ∗norm)
- HYPRE_Int HYPRE_StructLGMRESGetResidual (HYPRE_StructSolver solver, void ∗∗residual)

Struct BiCGSTAB Solver

These routines should be used in conjunction with the generic interface in \Ref{BiCGSTAB Solver}.

- HYPRE_Int HYPRE_StructBiCGSTABCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructBiCGSTABDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructBiCGSTABSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructBiCGSTABSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructBiCGSTABSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)

- HYPRE_Int HYPRE_StructBiCGSTABSetAbsoluteTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructBiCGSTABSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructBiCGSTABSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructBiCGSTABSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructBiCGSTABSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int level)
- HYPRE_Int HYPRE_StructBiCGSTABGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_↩ iterations)
- HYPRE_Int HYPRE_StructBiCGSTABGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE↩ _Real ∗norm)
- HYPRE_Int HYPRE_StructBiCGSTABGetResidual (HYPRE_StructSolver solver, void ∗∗residual)

Struct Hybrid Solver

- HYPRE_Int HYPRE_StructHybridCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructHybridDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructHybridSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructHybridSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructHybridSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructHybridSetConvergenceTol (HYPRE_StructSolver solver, HYPRE_Real cf_tol)
- HYPRE_Int HYPRE_StructHybridSetDSCGMaxIter (HYPRE_StructSolver solver, HYPRE_Int ds_max_its)
- HYPRE_Int HYPRE_StructHybridSetPCGMaxIter (HYPRE_StructSolver solver, HYPRE_Int pre_max_its)
- HYPRE_Int HYPRE_StructHybridSetTwoNorm (HYPRE_StructSolver solver, HYPRE_Int two_norm)
- HYPRE_Int HYPRE_StructHybridSetStopCrit (HYPRE_StructSolver solver, HYPRE_Int stop_crit)
- HYPRE_Int HYPRE_StructHybridSetRelChange (HYPRE_StructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_StructHybridSetSolverType (HYPRE_StructSolver solver, HYPRE_Int solver_type)
- HYPRE_Int HYPRE_StructHybridSetRecomputeResidual (HYPRE_StructSolver solver, HYPRE_Int recompute_residual)
- HYPRE_Int HYPRE_StructHybridGetRecomputeResidual (HYPRE_StructSolver solver, HYPRE_Int ∗recompute_residual)
- HYPRE_Int HYPRE_StructHybridSetRecomputeResidualP (HYPRE_StructSolver solver, HYPRE_Int recompute_residual_p)
- HYPRE_Int HYPRE_StructHybridGetRecomputeResidualP (HYPRE_StructSolver solver, HYPRE_Int ∗recompute_residual_p)
- HYPRE_Int HYPRE_StructHybridSetKDim (HYPRE_StructSolver solver, HYPRE_Int k_dim)
- HYPRE_Int HYPRE_StructHybridSetPrecond (HYPRE_StructSolver solver, HYPRE_PtrToStructSolverFcn precond, HYPRE_PtrToStructSolverFcn precond_setup, HYPRE_StructSolver precond_solver)
- HYPRE_Int HYPRE_StructHybridSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructHybridSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_StructHybridGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_its)
- HYPRE_Int HYPRE_StructHybridGetDSCGNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗ds_↩ num_its)
- HYPRE_Int HYPRE_StructHybridGetPCGNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗pre_↩ num_its)
- HYPRE_Int HYPRE_StructHybridGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)
- HYPRE_Int HYPRE_StructSparseMSGCreate (MPI_Comm comm, HYPRE_StructSolver ∗solver)
- HYPRE_Int HYPRE_StructSparseMSGDestroy (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSparseMSGSetup (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructSparseMSGSolve (HYPRE_StructSolver solver, HYPRE_StructMatrix A, HYPRE_StructVector b, HYPRE_StructVector x)
- HYPRE_Int HYPRE_StructSparseMSGSetTol (HYPRE_StructSolver solver, HYPRE_Real tol)
- HYPRE_Int HYPRE_StructSparseMSGSetMaxIter (HYPRE_StructSolver solver, HYPRE_Int max_iter)
- HYPRE_Int HYPRE_StructSparseMSGSetJump (HYPRE_StructSolver solver, HYPRE_Int jump)

- HYPRE_Int HYPRE_StructSparseMSGSetRelChange (HYPRE_StructSolver solver, HYPRE_Int rel_change)
- HYPRE_Int HYPRE_StructSparseMSGSetZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSparseMSGSetNonZeroGuess (HYPRE_StructSolver solver)
- HYPRE_Int HYPRE_StructSparseMSGSetRelaxType (HYPRE_StructSolver solver, HYPRE_Int relax_type)
- HYPRE_Int HYPRE_StructSparseMSGSetJacobiWeight (HYPRE_StructSolver solver, HYPRE_Real weight)
- HYPRE_Int HYPRE_StructSparseMSGSetNumPreRelax (HYPRE_StructSolver solver, HYPRE_Int num_↩ pre_relax)
- HYPRE_Int HYPRE_StructSparseMSGSetNumPostRelax (HYPRE_StructSolver solver, HYPRE_Int num_↩ post_relax)
- HYPRE_Int HYPRE_StructSparseMSGSetNumFineRelax (HYPRE_StructSolver solver, HYPRE_Int num_↩ fine_relax)
- HYPRE_Int HYPRE_StructSparseMSGSetLogging (HYPRE_StructSolver solver, HYPRE_Int logging)
- HYPRE_Int HYPRE_StructSparseMSGSetPrintLevel (HYPRE_StructSolver solver, HYPRE_Int print_level)
- HYPRE_Int HYPRE_StructSparseMSGGetNumIterations (HYPRE_StructSolver solver, HYPRE_Int ∗num_↩ iterations)
- HYPRE_Int HYPRE_StructSparseMSGGetFinalRelativeResidualNorm (HYPRE_StructSolver solver, HYPRE_Real ∗norm)

Struct LOBPCG Eigensolver

These routines should be used in conjunction with the generic interface in \Ref{LOBPCG Eigensolver}.

- HYPRE_Int HYPRE_StructSetupInterpreter (mv_InterfaceInterpreter ∗i)
- HYPRE_Int HYPRE_StructSetupMatvec (HYPRE_MatvecFunctions ∗mv)
- HYPRE_Int hypre_StructVectorSetRandomValues (hypre_StructVector ∗vector, HYPRE_Int seed)
- HYPRE_Int hypre_StructSetRandomValues (void ∗v, HYPRE_Int seed)

Struct Solvers

- #define HYPRE_MODIFYPC
- #define HYPRE_SOLVER_STRUCT
- typedef struct hypre_StructSolver_struct ∗ HYPRE_StructSolver
- typedef HYPRE_Int(∗ HYPRE_PtrToStructSolverFcn) (HYPRE_StructSolver, HYPRE_StructMatrix, HYPRE_StructVector, HYPRE_StructVector)
- typedef struct hypre_Solver_struct ∗ HYPRE_Solver
- typedef HYPRE_Int(∗ HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)

2.7.1 Macro Definition Documentation

2.7.1.1 HYPRE_MODIFYPC

```
#define HYPRE_MODIFYPC
```

2.7.1.2 HYPRE_SOLVER_STRUCT

```
#define HYPRE_SOLVER_STRUCT
```

2.7.2 Typedef Documentation

2.7.2.1 HYPRE_PtrToModifyPCFcn

```
typedef HYPRE_Int(* HYPRE_PtrToModifyPCFcn) (HYPRE_Solver, HYPRE_Int, HYPRE_Real)
```

2.7.2.2 HYPRE_PtrToStructSolverFcn

```
typedef HYPRE_Int(* HYPRE_PtrToStructSolverFcn) (HYPRE_StructSolver, HYPRE_StructMatrix, HYPRE_StructVector,
HYPRE_StructVector)
```

2.7.2.3 HYPRE_Solver

```
typedef struct hypre_Solver_struct* HYPRE_Solver
```

2.7.2.4 HYPRE_StructSolver

```
typedef struct hypre_StructSolver_struct* HYPRE_StructSolver
```

The solver object.

2.7.3 Function Documentation

2.7.3.1 HYPRE_StructBiCGSTABCreate()

```
HYPRE_Int HYPRE_StructBiCGSTABCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.2 HYPRE_StructBiCGSTABDestroy()

```
HYPRE_Int HYPRE_StructBiCGSTABDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.3 HYPRE_StructBiCGSTABGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructBiCGSTABGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.4 HYPRE_StructBiCGSTABGetNumIterations()

```
HYPRE_Int HYPRE_StructBiCGSTABGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.5 HYPRE_StructBiCGSTABGetResidual()

```
HYPRE_Int HYPRE_StructBiCGSTABGetResidual (
            HYPRE_StructSolver solver,
            void ** residual )
```

2.7.3.6 HYPRE_StructBiCGSTABSetAbsoluteTol()

```
HYPRE_Int HYPRE_StructBiCGSTABSetAbsoluteTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.7 HYPRE_StructBiCGSTABSetLogging()

```
HYPRE_Int HYPRE_StructBiCGSTABSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.8 HYPRE_StructBiCGSTABSetMaxIter()

```
HYPRE_Int HYPRE_StructBiCGSTABSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.9 HYPRE_StructBiCGSTABSetPrecond()

```
HYPRE_Int HYPRE_StructBiCGSTABSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

2.7.3.10 HYPRE_StructBiCGSTABSetPrintLevel()

```
HYPRE_Int HYPRE_StructBiCGSTABSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int level )
```

2.7.3.11 HYPRE_StructBiCGSTABSetTol()

```
HYPRE_Int HYPRE_StructBiCGSTABSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.12 HYPRE_StructBiCGSTABSetup()

```
HYPRE_Int HYPRE_StructBiCGSTABSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.13 HYPRE_StructBiCGSTABSolve()

```
HYPRE_Int HYPRE_StructBiCGSTABSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.14 HYPRE_StructCycRedCreate()

```
HYPRE_Int HYPRE_StructCycRedCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.15 HYPRE_StructCycRedDestroy()

```
HYPRE_Int HYPRE_StructCycRedDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.16 HYPRE_StructCycRedSetBase()

```
HYPRE_Int HYPRE_StructCycRedSetBase (
            HYPRE_StructSolver solver,
            HYPRE_Int ndim,
            HYPRE_Int * base_index,
            HYPRE_Int * base_stride )
```

(Optional) Set the base index and stride for the embedded 1D systems. The stride must be equal one in the dimension corresponding to the 1D systems (see \Ref{HYPRE_StructCycRedSetTDim}).

2.7.3.17 HYPRE_StructCycRedSetTDim()

```
HYPRE_Int HYPRE_StructCycRedSetTDim (
            HYPRE_StructSolver solver,
            HYPRE_Int tdim )
```

(Optional) Set the dimension number for the embedded 1D tridiagonal systems. The default is {\tt tdim} = 0.

2.7.3.18 HYPRE_StructCycRedSetup()

```
HYPRE_Int HYPRE_StructCycRedSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.7.3.19 HYPRE_StructCycRedSolve()

```
HYPRE_Int HYPRE_StructCycRedSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Solve the system.

2.7.3.20 HYPRE_StructDiagScale()

```
HYPRE_Int HYPRE_StructDiagScale (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix HA,
            HYPRE_StructVector Hy,
            HYPRE_StructVector Hx )
```

Solve routine for diagonal preconditioning.

2.7.3.21 HYPRE_StructDiagScaleSetup()

```
HYPRE_Int HYPRE_StructDiagScaleSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector y,
            HYPRE_StructVector x )
```

Setup routine for diagonal preconditioning.

2.7.3.22 HYPRE_StructFlexGMRESCreate()

```
HYPRE_Int HYPRE_StructFlexGMRESCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.23 HYPRE_StructFlexGMRESDestroy()

```
HYPRE_Int HYPRE_StructFlexGMRESDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.24 HYPRE_StructFlexGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructFlexGMRESGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.25 HYPRE_StructFlexGMRESGetNumIterations()

```
HYPRE_Int HYPRE_StructFlexGMRESGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.26 HYPRE_StructFlexGMRESGetResidual()

```
HYPRE_Int HYPRE_StructFlexGMRESGetResidual (
            HYPRE_StructSolver solver,
            void ** residual )
```

2.7.3.27 HYPRE_StructFlexGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_StructFlexGMRESSetAbsoluteTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.28 HYPRE_StructFlexGMRESSetKDim()

```
HYPRE_Int HYPRE_StructFlexGMRESSetKDim (
            HYPRE_StructSolver solver,
            HYPRE_Int k_dim )
```

2.7.3.29 HYPRE_StructFlexGMRESSetLogging()

```
HYPRE_Int HYPRE_StructFlexGMRESSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.30 HYPRE_StructFlexGMRESSetMaxIter()

```
HYPRE_Int HYPRE_StructFlexGMRESSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.31 HYPRE_StructFlexGMRESSetModifyPC()

```
HYPRE_Int HYPRE_StructFlexGMRESSetModifyPC (
            HYPRE_StructSolver solver,
            HYPRE_PtrToModifyPCFcn modify_pc )
```

2.7.3.32 HYPRE_StructFlexGMRESSetPrecond()

```
HYPRE_Int HYPRE_StructFlexGMRESSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

2.7.3.33 HYPRE_StructFlexGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_StructFlexGMRESSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int level )
```

2.7.3.34 HYPRE_StructFlexGMRESSetTol()

```
HYPRE_Int HYPRE_StructFlexGMRESSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.35 HYPRE_StructFlexGMRESSetup()

```
HYPRE_Int HYPRE_StructFlexGMRESSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.36 HYPRE_StructFlexGMRESSolve()

```
HYPRE_Int HYPRE_StructFlexGMRESSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.37 HYPRE_StructGMRESCreate()

```
HYPRE_Int HYPRE_StructGMRESCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.38 HYPRE_StructGMRESDestroy()

```
HYPRE_Int HYPRE_StructGMRESDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.39 HYPRE_StructGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructGMRESGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.40 HYPRE_StructGMRESGetNumIterations()

```
HYPRE_Int HYPRE_StructGMRESGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.41 HYPRE_StructGMRESGetResidual()

```
HYPRE_Int HYPRE_StructGMRESGetResidual (
            HYPRE_StructSolver solver,
            void ** residual )
```

2.7.3.42 HYPRE_StructGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_StructGMRESSetAbsoluteTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.43 HYPRE_StructGMRESSetKDim()

```
HYPRE_Int HYPRE_StructGMRESSetKDim (
            HYPRE_StructSolver solver,
            HYPRE_Int k_dim )
```

2.7.3.44 HYPRE_StructGMRESSetLogging()

```
HYPRE_Int HYPRE_StructGMRESSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.45 HYPRE_StructGMRESSetMaxIter()

```
HYPRE_Int HYPRE_StructGMRESSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.46 HYPRE_StructGMRESSetPrecond()

```
HYPRE_Int HYPRE_StructGMRESSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

2.7.3.47 HYPRE_StructGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_StructGMRESSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int level )
```

2.7.3.48 HYPRE_StructGMRESSetTol()

```
HYPRE_Int HYPRE_StructGMRESSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.49 HYPRE_StructGMRESSetup()

```
HYPRE_Int HYPRE_StructGMRESSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.50 HYPRE_StructGMRESSolve()

```
HYPRE_Int HYPRE_StructGMRESSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.51 HYPRE_StructHybridCreate()

```
HYPRE_Int HYPRE_StructHybridCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.52 HYPRE_StructHybridDestroy()

```
HYPRE_Int HYPRE_StructHybridDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.53 HYPRE_StructHybridGetDSCGNumIterations()

```
HYPRE_Int HYPRE_StructHybridGetDSCGNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * ds_num_its )
```

Return the number of diagonal scaling iterations taken.

2.7.3.54 HYPRE_StructHybridGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructHybridGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.7.3.55 HYPRE_StructHybridGetNumIterations()

```
HYPRE_Int HYPRE_StructHybridGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_its )
```

Return the number of iterations taken.

2.7.3.56 HYPRE_StructHybridGetPCGNumIterations()

```
HYPRE_Int HYPRE_StructHybridGetPCGNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * pre_num_its )
```

Return the number of general preconditioning iterations taken.

2.7.3.57 HYPRE_StructHybridGetRecomputeResidual()

```
HYPRE_Int HYPRE_StructHybridGetRecomputeResidual (
            HYPRE_StructSolver solver,
            HYPRE_Int * recompute_residual )
```

(Optional) Get recompute residual option.

2.7.3.58 HYPRE_StructHybridGetRecomputeResidualP()

```
HYPRE_Int HYPRE_StructHybridGetRecomputeResidualP (
            HYPRE_StructSolver solver,
            HYPRE_Int * recompute_residual_p )
```

(Optional) Get recompute residual period option.

2.7.3.59 HYPRE_StructHybridSetConvergenceTol()

```
HYPRE_Int HYPRE_StructHybridSetConvergenceTol (
            HYPRE_StructSolver solver,
            HYPRE_Real cf_tol )
```

(Optional) Set an accepted convergence tolerance for diagonal scaling (DS). The solver will switch preconditioners if the convergence of DS is slower than {\tt cf_tol}.

2.7.3.60 HYPRE_StructHybridSetDSCGMaxIter()

```
HYPRE_Int HYPRE_StructHybridSetDSCGMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int ds_max_its )
```

(Optional) Set maximum number of iterations for diagonal scaling (DS). The solver will switch preconditioners if DS reaches {\tt ds_max_its}.

2.7.3.61 HYPRE_StructHybridSetKDim()

```
HYPRE_Int HYPRE_StructHybridSetKDim (
            HYPRE_StructSolver solver,
            HYPRE_Int k_dim )
```

(Optional) Set the maximum size of the Krylov space when using GMRES.

2.7.3.62 HYPRE_StructHybridSetLogging()

```
HYPRE_Int HYPRE_StructHybridSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.7.3.63 HYPRE_StructHybridSetPCGMaxIter()

```
HYPRE_Int HYPRE_StructHybridSetPCGMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int pre_max_its )
```

(Optional) Set maximum number of iterations for general preconditioner (PRE). The solver will stop if PRE reaches {\tt pre_max_its}.

2.7.3.64 HYPRE_StructHybridSetPrecond()

```
HYPRE_Int HYPRE_StructHybridSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

(Optional) Set the preconditioner to use.

2.7.3.65 HYPRE_StructHybridSetPrintLevel()

```
HYPRE_Int HYPRE_StructHybridSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int print_level )
```

(Optional) Set the amount of printing to do to the screen.

2.7.3.66 HYPRE_StructHybridSetRecomputeResidual()

```
HYPRE_Int HYPRE_StructHybridSetRecomputeResidual (
            HYPRE_StructSolver solver,
            HYPRE_Int recompute_residual )
```

(Optional) Set recompute residual (don't rely on 3-term recurrence).

2.7.3.67 HYPRE_StructHybridSetRecomputeResidualP()

```
HYPRE_Int HYPRE_StructHybridSetRecomputeResidualP (
            HYPRE_StructSolver solver,
            HYPRE_Int recompute_residual_p )
```

(Optional) Set recompute residual period (don't rely on 3-term recurrence).

Recomputes residual after every specified number of iterations.

2.7.3.68 HYPRE_StructHybridSetRelChange()

```
HYPRE_Int HYPRE_StructHybridSetRelChange (
            HYPRE_StructSolver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.7.3.69 HYPRE_StructHybridSetSolverType()

```
HYPRE_Int HYPRE_StructHybridSetSolverType (
            HYPRE_StructSolver solver,
            HYPRE_Int solver_type )
```

(Optional) Set the type of Krylov solver to use.

Current krylov methods set by {\tt solver_type} are:

\begin{tabular}{l – }l} 0 & PCG (default) \ 1 & GMRES \ 2 & BiCGSTAB \ \end{tabular}

2.7.3.70 HYPRE_StructHybridSetStopCrit()

```
HYPRE_Int HYPRE_StructHybridSetStopCrit (
            HYPRE_StructSolver solver,
            HYPRE_Int stop_crit )
```

2.7.3.71 HYPRE_StructHybridSetTol()

```
HYPRE_Int HYPRE_StructHybridSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.7.3.72 HYPRE_StructHybridSetTwoNorm()

```
HYPRE_Int HYPRE_StructHybridSetTwoNorm (
            HYPRE_StructSolver solver,
            HYPRE_Int two_norm )
```

(Optional) Use the two-norm in stopping criteria.

2.7.3.73 HYPRE_StructHybridSetup()

```
HYPRE_Int HYPRE_StructHybridSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.7.3.74 HYPRE_StructHybridSolve()

```
HYPRE_Int HYPRE_StructHybridSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Solve the system.

2.7.3.75 HYPRE_StructJacobiCreate()

```
HYPRE_Int HYPRE_StructJacobiCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.76 HYPRE_StructJacobiDestroy()

```
HYPRE_Int HYPRE_StructJacobiDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.7.3.77 HYPRE_StructJacobiGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructJacobiGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.7.3.78 HYPRE_StructJacobiGetNumIterations()

```
HYPRE_Int HYPRE_StructJacobiGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.7.3.79 HYPRE_StructJacobiSetMaxIter()

```
HYPRE_Int HYPRE_StructJacobiSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.7.3.80 HYPRE_StructJacobiSetNonZeroGuess()

```
HYPRE_Int HYPRE_StructJacobiSetNonZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.7.3.81 HYPRE_StructJacobiSetTol()

```
HYPRE_Int HYPRE_StructJacobiSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.7.3.82 HYPRE_StructJacobiSetup()

```
HYPRE_Int HYPRE_StructJacobiSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.7.3.83 HYPRE_StructJacobiSetZeroGuess()

```
HYPRE_Int HYPRE_StructJacobiSetZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.7.3.84 HYPRE_StructJacobiSolve()

```
HYPRE_Int HYPRE_StructJacobiSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Solve the system.

2.7.3.85 HYPRE_StructLGMRESCreate()

```
HYPRE_Int HYPRE_StructLGMRESCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.86 HYPRE_StructLGMRESDestroy()

```
HYPRE_Int HYPRE_StructLGMRESDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.87 HYPRE_StructLGMRESGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructLGMRESGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.88 HYPRE_StructLGMRESGetNumIterations()

```
HYPRE_Int HYPRE_StructLGMRESGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.89 HYPRE_StructLGMRESGetResidual()

```
HYPRE_Int HYPRE_StructLGMRESGetResidual (
            HYPRE_StructSolver solver,
            void ** residual )
```

2.7.3.90 HYPRE_StructLGMRESSetAbsoluteTol()

```
HYPRE_Int HYPRE_StructLGMRESSetAbsoluteTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.91 HYPRE_StructLGMRESSetAugDim()

```
HYPRE_Int HYPRE_StructLGMRESSetAugDim (
            HYPRE_StructSolver solver,
            HYPRE_Int aug_dim )
```

2.7.3.92 HYPRE_StructLGMRESSetKDim()

```
HYPRE_Int HYPRE_StructLGMRESSetKDim (
            HYPRE_StructSolver solver,
            HYPRE_Int k_dim )
```

2.7.3.93 HYPRE_StructLGMRESSetLogging()

```
HYPRE_Int HYPRE_StructLGMRESSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.94 HYPRE_StructLGMRESSetMaxIter()

```
HYPRE_Int HYPRE_StructLGMRESSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.95 HYPRE_StructLGMRESSetPrecond()

```
HYPRE_Int HYPRE_StructLGMRESSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

2.7.3.96 HYPRE_StructLGMRESSetPrintLevel()

```
HYPRE_Int HYPRE_StructLGMRESSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int level )
```

2.7.3.97 HYPRE_StructLGMRESSetTol()

```
HYPRE_Int HYPRE_StructLGMRESSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.98 HYPRE_StructLGMRESSetup()

```
HYPRE_Int HYPRE_StructLGMRESSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.99 HYPRE_StructLGMRESSolve()

```
HYPRE_Int HYPRE_StructLGMRESSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.100 HYPRE_StructPCGCreate()

```
HYPRE_Int HYPRE_StructPCGCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.101 HYPRE_StructPCGDestroy()

```
HYPRE_Int HYPRE_StructPCGDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.102 HYPRE_StructPCGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructPCGGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.103 HYPRE_StructPCGGetNumIterations()

```
HYPRE_Int HYPRE_StructPCGGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.104 HYPRE_StructPCGGetResidual()

```
HYPRE_Int HYPRE_StructPCGGetResidual (
            HYPRE_StructSolver solver,
            void ** residual )
```

2.7.3.105 HYPRE_StructPCGSetAbsoluteTol()

```
HYPRE_Int HYPRE_StructPCGSetAbsoluteTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.106 HYPRE_StructPCGSetLogging()

```
HYPRE_Int HYPRE_StructPCGSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.107 HYPRE_StructPCGSetMaxIter()

```
HYPRE_Int HYPRE_StructPCGSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.108 HYPRE_StructPCGSetPrecond()

```
HYPRE_Int HYPRE_StructPCGSetPrecond (
            HYPRE_StructSolver solver,
            HYPRE_PtrToStructSolverFcn precond,
            HYPRE_PtrToStructSolverFcn precond_setup,
            HYPRE_StructSolver precond_solver )
```

2.7.3.109 HYPRE_StructPCGSetPrintLevel()

```
HYPRE_Int HYPRE_StructPCGSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int level )
```

2.7.3.110 HYPRE_StructPCGSetRelChange()

```
HYPRE_Int HYPRE_StructPCGSetRelChange (
            HYPRE_StructSolver solver,
            HYPRE_Int rel_change )
```

2.7.3.111 HYPRE_StructPCGSetTol()

```
HYPRE_Int HYPRE_StructPCGSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.112 HYPRE_StructPCGSetTwoNorm()

```
HYPRE_Int HYPRE_StructPCGSetTwoNorm (
            HYPRE_StructSolver solver,
            HYPRE_Int two_norm )
```

2.7.3.113 HYPRE_StructPCGSetup()

```
HYPRE_Int HYPRE_StructPCGSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.114 HYPRE_StructPCGSolve()

```
HYPRE_Int HYPRE_StructPCGSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.115 HYPRE_StructPFMGCreate()

```
HYPRE_Int HYPRE_StructPFMGCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.116 HYPRE_StructPFMGDestroy()

```
HYPRE_Int HYPRE_StructPFMGDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.117 HYPRE_StructPFMGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructPFMGGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.7.3.118 HYPRE_StructPFMGGetJacobiWeight()

```
HYPRE_Int HYPRE_StructPFMGGetJacobiWeight (
            HYPRE_StructSolver solver,
            HYPRE_Real * weight )
```

2.7.3.119 HYPRE_StructPFMGGetNumIterations()

```
HYPRE_Int HYPRE_StructPFMGGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.7.3.120 HYPRE_StructPFMGSetDxyz()

```
HYPRE_Int HYPRE_StructPFMGSetDxyz (
            HYPRE_StructSolver solver,
            HYPRE_Real * dxyz )
```

2.7.3.121 HYPRE_StructPFMGSetJacobiWeight()

```
HYPRE_Int HYPRE_StructPFMGSetJacobiWeight (
            HYPRE_StructSolver solver,
            HYPRE_Real weight )
```

2.7.3.122 HYPRE_StructPFMGSetLogging()

```
HYPRE_Int HYPRE_StructPFMGSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.7.3.123 HYPRE_StructPFMGSetMaxIter()

```
HYPRE_Int HYPRE_StructPFMGSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.7.3.124 HYPRE_StructPFMGSetMaxLevels()

```
HYPRE_Int HYPRE_StructPFMGSetMaxLevels (
            HYPRE_StructSolver solver,
            HYPRE_Int max_levels )
```

(Optional) Set maximum number of multigrid grid levels.

2.7.3.125 HYPRE_StructPFMGSetNonZeroGuess()

```
HYPRE_Int HYPRE_StructPFMGSetNonZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.7.3.126 HYPRE_StructPFMGSetNumPostRelax()

```
HYPRE_Int HYPRE_StructPFMGSetNumPostRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_post_relax )
```

(Optional) Set number of relaxation sweeps after coarse-grid correction.

2.7.3.127 HYPRE_StructPFMGSetNumPreRelax()

```
HYPRE_Int HYPRE_StructPFMGSetNumPreRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_pre_relax )
```

(Optional) Set number of relaxation sweeps before coarse-grid correction.

2.7.3.128 HYPRE_StructPFMGSetPrintLevel()

```
HYPRE_Int HYPRE_StructPFMGSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int print_level )
```

(Optional) Set the amount of printing to do to the screen.

2.7.3.129 HYPRE_StructPFMGSetRAPType()

```
HYPRE_Int HYPRE_StructPFMGSetRAPType (
            HYPRE_StructSolver solver,
            HYPRE_Int rap_type )
```

(Optional) Set type of coarse-grid operator to use.

Current operators set by {\tt rap_type} are:

\begin{tabular}{l – }l} 0 & Galerkin (default) \ 1 & non-Galerkin 5-pt or 7-pt stencils \ \end{tabular}

Both operators are constructed algebraically. The non-Galerkin option maintains a 5-pt stencil in 2D and a 7-pt stencil in 3D on all grid levels. The stencil coefficients are computed by averaging techniques.

2.7.3.130 HYPRE_StructPFMGSetRelaxType()

```
HYPRE_Int HYPRE_StructPFMGSetRelaxType (
            HYPRE_StructSolver solver,
            HYPRE_Int relax_type )
```

(Optional) Set relaxation type.

Current relaxation methods set by {\tt relax_type} are:

\begin{tabular}{l – }l} 0 & Jacobi \ 1 & Weighted Jacobi (default) \ 2 & Red/Black Gauss-Seidel (symmetric: RB pre-relaxation, BR post-relaxation) \ 3 & Red/Black Gauss-Seidel (nonsymmetric: RB pre- and post-relaxation) \ \end{tabular}

2.7.3.131 HYPRE_StructPFMGSetRelChange()

```
HYPRE_Int HYPRE_StructPFMGSetRelChange (
            HYPRE_StructSolver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.7.3.132 HYPRE_StructPFMGSetSkipRelax()

```
HYPRE_Int HYPRE_StructPFMGSetSkipRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int skip_relax )
```

(Optional) Skip relaxation on certain grids for isotropic problems. This can greatly improve efficiency by eliminating unnecessary relaxations when the underlying problem is isotropic.

2.7.3.133 HYPRE_StructPFMGSetTol()

```
HYPRE_Int HYPRE_StructPFMGSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.7.3.134 HYPRE_StructPFMGSetup()

```
HYPRE_Int HYPRE_StructPFMGSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.7.3.135 HYPRE_StructPFMGSetZeroGuess()

```
HYPRE_Int HYPRE_StructPFMGSetZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.7.3.136 HYPRE_StructPFMGSolve()

```
HYPRE_Int HYPRE_StructPFMGSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Solve the system.

2.7.3.137 hypre_StructSetRandomValues()

```
HYPRE_Int hypre_StructSetRandomValues (
            void * v,
            HYPRE_Int seed )
```

2.7.3.138 HYPRE_StructSetupInterpreter()

```
HYPRE_Int HYPRE_StructSetupInterpreter (
            mv_InterfaceInterpreter * i )
```

Load interface interpreter. Vector part loaded with hypre_StructKrylov functions and multivector part loaded with mv_↩
TempMultiVector functions.

2.7.3.139 HYPRE_StructSetupMatvec()

```
HYPRE_Int HYPRE_StructSetupMatvec (
            HYPRE_MatvecFunctions * mv )
```

Load Matvec interpreter with hypre_StructKrylov functions.

2.7.3.140 HYPRE_StructSMGCreate()

```
HYPRE_Int HYPRE_StructSMGCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

Create a solver object.

2.7.3.141 HYPRE_StructSMGDestroy()

```
HYPRE_Int HYPRE_StructSMGDestroy (
            HYPRE_StructSolver solver )
```

Destroy a solver object.

2.7.3.142 HYPRE_StructSMGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructSMGGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

Return the norm of the final relative residual.

2.7.3.143 HYPRE_StructSMGGetNumIterations()

```
HYPRE_Int HYPRE_StructSMGGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

Return the number of iterations taken.

2.7.3.144 HYPRE_StructSMGSetLogging()

```
HYPRE_Int HYPRE_StructSMGSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

(Optional) Set the amount of logging to do.

2.7.3.145 HYPRE_StructSMGSetMaxIter()

```
HYPRE_Int HYPRE_StructSMGSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

(Optional) Set maximum number of iterations.

2.7.3.146 HYPRE_StructSMGSetMemoryUse()

```
HYPRE_Int HYPRE_StructSMGSetMemoryUse (
            HYPRE_StructSolver solver,
            HYPRE_Int memory_use )
```

2.7.3.147 HYPRE_StructSMGSetNonZeroGuess()

```
HYPRE_Int HYPRE_StructSMGSetNonZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a nonzero initial guess. This is the default behavior, but this routine allows the user to switch back after using {\tt SetZeroGuess}.

2.7.3.148 HYPRE_StructSMGSetNumPostRelax()

```
HYPRE_Int HYPRE_StructSMGSetNumPostRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_post_relax )
```

(Optional) Set number of relaxation sweeps after coarse-grid correction.

2.7.3.149 HYPRE_StructSMGSetNumPreRelax()

```
HYPRE_Int HYPRE_StructSMGSetNumPreRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_pre_relax )
```

(Optional) Set number of relaxation sweeps before coarse-grid correction.

2.7.3.150 HYPRE_StructSMGSetPrintLevel()

```
HYPRE_Int HYPRE_StructSMGSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int print_level )
```

(Optional) Set the amount of printing to do to the screen.

2.7.3.151 HYPRE_StructSMGSetRelChange()

```
HYPRE_Int HYPRE_StructSMGSetRelChange (
            HYPRE_StructSolver solver,
            HYPRE_Int rel_change )
```

(Optional) Additionally require that the relative difference in successive iterates be small.

2.7.3.152 HYPRE_StructSMGSetTol()

```
HYPRE_Int HYPRE_StructSMGSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

(Optional) Set the convergence tolerance.

2.7.3.153 HYPRE_StructSMGSetup()

```
HYPRE_Int HYPRE_StructSMGSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Prepare to solve the system. The coefficient data in {\tt b} and {\tt x} is ignored here, but information about the layout of the data may be used.

2.7.3.154 HYPRE_StructSMGSetZeroGuess()

```
HYPRE_Int HYPRE_StructSMGSetZeroGuess (
            HYPRE_StructSolver solver )
```

(Optional) Use a zero initial guess. This allows the solver to cut corners in the case where a zero initial guess is needed (e.g., for preconditioning) to reduce compuational cost.

2.7.3.155 HYPRE_StructSMGSolve()

```
HYPRE_Int HYPRE_StructSMGSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

Solve the system.

2.7.3.156 HYPRE_StructSparseMSGCreate()

```
HYPRE_Int HYPRE_StructSparseMSGCreate (
            MPI_Comm comm,
            HYPRE_StructSolver * solver )
```

2.7.3.157 HYPRE_StructSparseMSGDestroy()

```
HYPRE_Int HYPRE_StructSparseMSGDestroy (
            HYPRE_StructSolver solver )
```

2.7.3.158 HYPRE_StructSparseMSGGetFinalRelativeResidualNorm()

```
HYPRE_Int HYPRE_StructSparseMSGGetFinalRelativeResidualNorm (
            HYPRE_StructSolver solver,
            HYPRE_Real * norm )
```

2.7.3.159 HYPRE_StructSparseMSGGetNumIterations()

```
HYPRE_Int HYPRE_StructSparseMSGGetNumIterations (
            HYPRE_StructSolver solver,
            HYPRE_Int * num_iterations )
```

2.7.3.160 HYPRE_StructSparseMSGSetJacobiWeight()

```
HYPRE_Int HYPRE_StructSparseMSGSetJacobiWeight (
            HYPRE_StructSolver solver,
            HYPRE_Real weight )
```

2.7.3.161 HYPRE_StructSparseMSGSetJump()

```
HYPRE_Int HYPRE_StructSparseMSGSetJump (
            HYPRE_StructSolver solver,
            HYPRE_Int jump )
```

2.7.3.162 HYPRE_StructSparseMSGSetLogging()

```
HYPRE_Int HYPRE_StructSparseMSGSetLogging (
            HYPRE_StructSolver solver,
            HYPRE_Int logging )
```

2.7.3.163 HYPRE_StructSparseMSGSetMaxIter()

```
HYPRE_Int HYPRE_StructSparseMSGSetMaxIter (
            HYPRE_StructSolver solver,
            HYPRE_Int max_iter )
```

2.7.3.164 HYPRE_StructSparseMSGSetNonZeroGuess()

```
HYPRE_Int HYPRE_StructSparseMSGSetNonZeroGuess (
            HYPRE_StructSolver solver )
```

2.7.3.165 HYPRE_StructSparseMSGSetNumFineRelax()

```
HYPRE_Int HYPRE_StructSparseMSGSetNumFineRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_fine_relax )
```

2.7.3.166 HYPRE_StructSparseMSGSetNumPostRelax()

```
HYPRE_Int HYPRE_StructSparseMSGSetNumPostRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_post_relax )
```

2.7.3.167 HYPRE_StructSparseMSGSetNumPreRelax()

```
HYPRE_Int HYPRE_StructSparseMSGSetNumPreRelax (
            HYPRE_StructSolver solver,
            HYPRE_Int num_pre_relax )
```

2.7.3.168 HYPRE_StructSparseMSGSetPrintLevel()

```
HYPRE_Int HYPRE_StructSparseMSGSetPrintLevel (
            HYPRE_StructSolver solver,
            HYPRE_Int print_level )
```

2.7.3.169 HYPRE_StructSparseMSGSetRelaxType()

```
HYPRE_Int HYPRE_StructSparseMSGSetRelaxType (
            HYPRE_StructSolver solver,
            HYPRE_Int relax_type )
```

2.7.3.170 HYPRE_StructSparseMSGSetRelChange()

```
HYPRE_Int HYPRE_StructSparseMSGSetRelChange (
            HYPRE_StructSolver solver,
            HYPRE_Int rel_change )
```

2.7.3.171 HYPRE_StructSparseMSGSetTol()

```
HYPRE_Int HYPRE_StructSparseMSGSetTol (
            HYPRE_StructSolver solver,
            HYPRE_Real tol )
```

2.7.3.172 HYPRE_StructSparseMSGSetup()

```
HYPRE_Int HYPRE_StructSparseMSGSetup (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.173 HYPRE_StructSparseMSGSetZeroGuess()

```
HYPRE_Int HYPRE_StructSparseMSGSetZeroGuess (
            HYPRE_StructSolver solver )
```

2.7.3.174 HYPRE_StructSparseMSGSolve()

```
HYPRE_Int HYPRE_StructSparseMSGSolve (
            HYPRE_StructSolver solver,
            HYPRE_StructMatrix A,
            HYPRE_StructVector b,
            HYPRE_StructVector x )
```

2.7.3.175 hypre_StructVectorSetRandomValues()

```
HYPRE_Int hypre_StructVectorSetRandomValues (
            hypre_StructVector * vector,
            HYPRE_Int seed )
```

2.8 HYPRE_struct_mv.h File Reference

Macros

- #define HYPRE_StructVector_defined

Typedefs

- typedef struct hypre_StructVector_struct ∗ HYPRE_StructVector

Struct Grids

- typedef struct hypre_StructGrid_struct ∗ HYPRE_StructGrid
- HYPRE_Int HYPRE_StructGridCreate (MPI_Comm comm, HYPRE_Int ndim, HYPRE_StructGrid ∗grid)
- HYPRE_Int HYPRE_StructGridDestroy (HYPRE_StructGrid grid)
- HYPRE_Int HYPRE_StructGridSetExtents (HYPRE_StructGrid grid, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper)
- HYPRE_Int HYPRE_StructGridAssemble (HYPRE_StructGrid grid)
- HYPRE_Int HYPRE_StructGridSetPeriodic (HYPRE_StructGrid grid, HYPRE_Int ∗periodic)
- HYPRE_Int HYPRE_StructGridSetNumGhost (HYPRE_StructGrid grid, HYPRE_Int ∗num_ghost)

Struct Stencils

- typedef struct hypre_StructStencil_struct * HYPRE_StructStencil
- HYPRE_Int HYPRE_StructStencilCreate (HYPRE_Int ndim, HYPRE_Int size, HYPRE_StructStencil *stencil)
- HYPRE_Int HYPRE_StructStencilDestroy (HYPRE_StructStencil stencil)
- HYPRE_Int HYPRE_StructStencilSetElement (HYPRE_StructStencil stencil, HYPRE_Int entry, HYPRE_Int *offset)

Struct Matrices

- typedef struct hypre_StructMatrix_struct * HYPRE_StructMatrix
- HYPRE_Int HYPRE_StructMatrixCreate (MPI_Comm comm, HYPRE_StructGrid grid, HYPRE_StructStencil stencil, HYPRE_StructMatrix *matrix)
- HYPRE_Int HYPRE_StructMatrixDestroy (HYPRE_StructMatrix matrix)
- HYPRE_Int HYPRE_StructMatrixInitialize (HYPRE_StructMatrix matrix)
- HYPRE_Int HYPRE_StructMatrixSetValues (HYPRE_StructMatrix matrix, HYPRE_Int *index, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixAddToValues (HYPRE_StructMatrix matrix, HYPRE_Int *index, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixSetConstantValues (HYPRE_StructMatrix matrix, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixAddToConstantValues (HYPRE_StructMatrix matrix, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixSetBoxValues (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixAddToBoxValues (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixSetBoxValues2 (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Int *vilower, HYPRE_Int *viupper, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixAddToBoxValues2 (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Int *vilower, HYPRE_Int *viupper, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixAssemble (HYPRE_StructMatrix matrix)
- HYPRE_Int HYPRE_StructMatrixGetValues (HYPRE_StructMatrix matrix, HYPRE_Int *index, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixGetBoxValues (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixGetBoxValues2 (HYPRE_StructMatrix matrix, HYPRE_Int *ilower, HYPRE_Int *iupper, HYPRE_Int nentries, HYPRE_Int *entries, HYPRE_Int *vilower, HYPRE_Int *viupper, HYPRE_Complex *values)
- HYPRE_Int HYPRE_StructMatrixSetSymmetric (HYPRE_StructMatrix matrix, HYPRE_Int symmetric)
- HYPRE_Int HYPRE_StructMatrixSetConstantEntries (HYPRE_StructMatrix matrix, HYPRE_Int nentries, HYPRE_Int *entries)
- HYPRE_Int HYPRE_StructMatrixSetNumGhost (HYPRE_StructMatrix matrix, HYPRE_Int *num_ghost)
- HYPRE_Int HYPRE_StructMatrixPrint (const char *filename, HYPRE_StructMatrix matrix, HYPRE_Int all)
- HYPRE_Int HYPRE_StructMatrixMatvec (HYPRE_Complex alpha, HYPRE_StructMatrix A, HYPRE_StructVector x, HYPRE_Complex beta, HYPRE_StructVector y)

Struct Vectors

- typedef struct hypre_CommPkg_struct ∗ HYPRE_CommPkg
- HYPRE_Int HYPRE_StructVectorCreate (MPI_Comm comm, HYPRE_StructGrid grid, HYPRE_StructVector ∗vector)
- HYPRE_Int HYPRE_StructVectorDestroy (HYPRE_StructVector vector)
- HYPRE_Int HYPRE_StructVectorInitialize (HYPRE_StructVector vector)
- HYPRE_Int HYPRE_StructVectorSetValues (HYPRE_StructVector vector, HYPRE_Int ∗index, HYPRE_Complex value)
- HYPRE_Int HYPRE_StructVectorAddToValues (HYPRE_StructVector vector, HYPRE_Int ∗index, HYPRE_↩ Complex value)
- HYPRE_Int HYPRE_StructVectorSetBoxValues (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorAddToBoxValues (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE↩ _Int ∗iupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorSetBoxValues2 (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorAddToBoxValues2 (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE↩ _Int ∗iupper, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorAssemble (HYPRE_StructVector vector)
- HYPRE_Int HYPRE_StructVectorGetValues (HYPRE_StructVector vector, HYPRE_Int ∗index, HYPRE_Complex ∗value)
- HYPRE_Int HYPRE_StructVectorGetBoxValues (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorGetBoxValues2 (HYPRE_StructVector vector, HYPRE_Int ∗ilower, HYPRE_Int ∗iupper, HYPRE_Int ∗vilower, HYPRE_Int ∗viupper, HYPRE_Complex ∗values)
- HYPRE_Int HYPRE_StructVectorPrint (const char ∗filename, HYPRE_StructVector vector, HYPRE_Int all)
- HYPRE_Int HYPRE_StructMatrixGetGrid (HYPRE_StructMatrix matrix, HYPRE_StructGrid ∗grid)
- HYPRE_Int HYPRE_StructVectorSetNumGhost (HYPRE_StructVector vector, HYPRE_Int ∗num_ghost)
- HYPRE_Int HYPRE_StructVectorSetConstantValues (HYPRE_StructVector vector, HYPRE_Complex values)
- HYPRE_Int HYPRE_StructVectorGetMigrateCommPkg (HYPRE_StructVector from_vector, HYPRE_StructVector to_vector, HYPRE_CommPkg ∗comm_pkg)
- HYPRE_Int HYPRE_StructVectorMigrate (HYPRE_CommPkg comm_pkg, HYPRE_StructVector from_vector, HYPRE_StructVector to_vector)
- HYPRE_Int HYPRE_CommPkgDestroy (HYPRE_CommPkg comm_pkg)

2.8.1 Macro Definition Documentation

2.8.1.1 HYPRE_StructVector_defined

```
#define HYPRE_StructVector_defined
```

2.8.2 Typedef Documentation

2.8.2.1 HYPRE_CommPkg

```
typedef struct hypre_CommPkg_struct* HYPRE_CommPkg
```

2.8.2.2 HYPRE_StructGrid

```
typedef struct hypre_StructGrid_struct* HYPRE_StructGrid
```

A grid object is constructed out of several `‘boxes'', defined on a global abstract index space.

2.8.2.3 HYPRE_StructMatrix

```
typedef struct hypre_StructMatrix_struct* HYPRE_StructMatrix
```

The matrix object.

2.8.2.4 HYPRE_StructStencil

```
typedef struct hypre_StructStencil_struct* HYPRE_StructStencil
```

The stencil object.

2.8.2.5 HYPRE_StructVector

```
typedef struct hypre_StructVector_struct* HYPRE_StructVector
```

2.8.3 Function Documentation

2.8.3.1 HYPRE_CommPkgDestroy()

```
HYPRE_Int HYPRE_CommPkgDestroy (
            HYPRE_CommPkg comm_pkg )
```

2.8.3.2 HYPRE_StructGridAssemble()

```
HYPRE_Int HYPRE_StructGridAssemble (
            HYPRE_StructGrid grid )
```

Finalize the construction of the grid before using.

2.8.3.3 HYPRE_StructGridCreate()

```
HYPRE_Int HYPRE_StructGridCreate (
            MPI_Comm comm,
            HYPRE_Int ndim,
            HYPRE_StructGrid * grid )
```

Create an {\tt ndim}-dimensional grid object.

2.8.3.4 HYPRE_StructGridDestroy()

```
HYPRE_Int HYPRE_StructGridDestroy (
            HYPRE_StructGrid grid )
```

Destroy a grid object. An object should be explicitly destroyed using this destructor when the user's code no longer needs direct access to it. Once destroyed, the object must not be referenced again. Note that the object may not be deallocated at the completion of this call, since there may be internal package references to the object. The object will then be destroyed when all internal reference counts go to zero.

2.8.3.5 HYPRE_StructGridSetExtents()

```
HYPRE_Int HYPRE_StructGridSetExtents (
            HYPRE_StructGrid grid,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper )
```

Set the extents for a box on the grid.

2.8.3.6 HYPRE_StructGridSetNumGhost()

```
HYPRE_Int HYPRE_StructGridSetNumGhost (
            HYPRE_StructGrid grid,
            HYPRE_Int * num_ghost )
```

Set the ghost layer in the grid object

2.8.3.7 HYPRE_StructGridSetPeriodic()

```
HYPRE_Int HYPRE_StructGridSetPeriodic (
            HYPRE_StructGrid grid,
            HYPRE_Int * periodic )
```

Set the periodicity for the grid.

The argument {\tt periodic} is an {\tt ndim}-dimensional integer array that contains the periodicity for each dimension. A zero value for a dimension means non-periodic, while a nonzero value means periodic and contains the actual period. For example, periodicity in the first and third dimensions for a 10x11x12 grid is indicated by the array [10,0,12].

NOTE: Some of the solvers in hypre have power-of-two restrictions on the size of the periodic dimensions.

2.8.3.8 HYPRE_StructMatrixAddToBoxValues()

```
HYPRE_Int HYPRE_StructMatrixAddToBoxValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Add to matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructMatrixSetBox↩ Values}.

2.8.3.9 HYPRE_StructMatrixAddToBoxValues2()

```
HYPRE_Int HYPRE_StructMatrixAddToBoxValues2 (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Add to matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructMatrixSetBox↩ Values2}.

2.8.3.10 HYPRE_StructMatrixAddToConstantValues()

```
HYPRE_Int HYPRE_StructMatrixAddToConstantValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Add to matrix coefficients which are constant over the grid. The {\tt values} array is of length {\tt nentries}.

2.8.3.11 HYPRE_StructMatrixAddToValues()

```
HYPRE_Int HYPRE_StructMatrixAddToValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * index,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Add to matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_StructMatrixAddToBoxValues} to set coefficients a box at a time.

2.8.3.12 HYPRE_StructMatrixAssemble()

```
HYPRE_Int HYPRE_StructMatrixAssemble (
            HYPRE_StructMatrix matrix )
```

Finalize the construction of the matrix before using.

2.8.3.13 HYPRE_StructMatrixCreate()

```
HYPRE_Int HYPRE_StructMatrixCreate (
            MPI_Comm comm,
            HYPRE_StructGrid grid,
            HYPRE_StructStencil stencil,
            HYPRE_StructMatrix * matrix )
```

Create a matrix object.

2.8.3.14 HYPRE_StructMatrixDestroy()

```
HYPRE_Int HYPRE_StructMatrixDestroy (
            HYPRE_StructMatrix matrix )
```

Destroy a matrix object.

2.8.3.15 HYPRE_StructMatrixGetBoxValues()

```
HYPRE_Int HYPRE_StructMatrixGetBoxValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Get matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructMatrixSetBoxValues}.

2.8.3.16 HYPRE_StructMatrixGetBoxValues2()

```
HYPRE_Int HYPRE_StructMatrixGetBoxValues2 (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Get matrix coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructMatrixSetBox←Values2}.

2.8.3.17 HYPRE_StructMatrixGetGrid()

```
HYPRE_Int HYPRE_StructMatrixGetGrid (
            HYPRE_StructMatrix matrix,
            HYPRE_StructGrid * grid )
```

2.8.3.18 HYPRE_StructMatrixGetValues()

```
HYPRE_Int HYPRE_StructMatrixGetValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * index,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Get matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_StructMatrixGetBoxValues} to get coefficients a box at a time.

2.8.3.19 HYPRE_StructMatrixInitialize()

```
HYPRE_Int HYPRE_StructMatrixInitialize (
            HYPRE_StructMatrix matrix )
```

Prepare a matrix object for setting coefficient values.

2.8.3.20 HYPRE_StructMatrixMatvec()

```
HYPRE_Int HYPRE_StructMatrixMatvec (
            HYPRE_Complex alpha,
            HYPRE_StructMatrix A,
            HYPRE_StructVector x,
            HYPRE_Complex beta,
            HYPRE_StructVector y )
```

Matvec operator. This operation is $y = \alpha A x + \beta y$. Note that you can do a simple matrix-vector multiply by setting $\alpha=1$ and $\beta=0$.

2.8.3.21 HYPRE_StructMatrixPrint()

```
HYPRE_Int HYPRE_StructMatrixPrint (
            const char * filename,
            HYPRE_StructMatrix matrix,
            HYPRE_Int all )
```

Print the matrix to file. This is mainly for debugging purposes.

2.8.3.22 HYPRE_StructMatrixSetBoxValues()

```
HYPRE_Int HYPRE_StructMatrixSetBoxValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Set matrix coefficients a box at a time. The data in {\tt values} is ordered as follows:

\begin{verbatim} m = 0; for (k = ilower[2]; k <= iupper[2]; k++) for (j = ilower[1]; j <= iupper[1]; j++) for (i = ilower[0]; i <= iupper[0]; i++) for (entry = 0; entry < nentries; entry++) { values[m] = ...; m++; } \end{verbatim}

2.8.3.23 HYPRE_StructMatrixSetBoxValues2()

```
HYPRE_Int HYPRE_StructMatrixSetBoxValues2 (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Set matrix coefficients a box at a time. The {\tt values} array is logically box shaped with value-box extents {\tt vilower} and {\tt viupper} that must contain the set-box extents {\tt ilower} and {\tt iupper} . The data in the {\tt values} array is ordered as in \Ref{HYPRE_StructMatrixSetBoxValues}, but based on the value-box extents.

2.8.3.24 HYPRE_StructMatrixSetConstantEntries()

```
HYPRE_Int HYPRE_StructMatrixSetConstantEntries (
            HYPRE_StructMatrix matrix,
            HYPRE_Int nentries,
            HYPRE_Int * entries )
```

Specify which stencil entries are constant over the grid. Declaring entries to be `‘constant over the grid'' yields significant memory savings because the value for each declared entry will only be stored once. However, not all solvers are able to utilize this feature.

Presently supported: \begin{itemize} \item no entries constant (this function need not be called) \item all entries constant \item all but the diagonal entry constant \end{itemize}

2.8.3.25 HYPRE_StructMatrixSetConstantValues()

```
HYPRE_Int HYPRE_StructMatrixSetConstantValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Set matrix coefficients which are constant over the grid. The {\tt values} array is of length {\tt nentries}.

2.8.3.26 HYPRE_StructMatrixSetNumGhost()

```
HYPRE_Int HYPRE_StructMatrixSetNumGhost (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * num_ghost )
```

Set the ghost layer in the matrix

2.8.3.27 HYPRE_StructMatrixSetSymmetric()

```
HYPRE_Int HYPRE_StructMatrixSetSymmetric (
            HYPRE_StructMatrix matrix,
            HYPRE_Int symmetric )
```

Define symmetry properties of the matrix. By default, matrices are assumed to be nonsymmetric. Significant storage savings can be made if the matrix is symmetric.

2.8.3.28 HYPRE_StructMatrixSetValues()

```
HYPRE_Int HYPRE_StructMatrixSetValues (
            HYPRE_StructMatrix matrix,
            HYPRE_Int * index,
            HYPRE_Int nentries,
            HYPRE_Int * entries,
            HYPRE_Complex * values )
```

Set matrix coefficients index by index. The {\tt values} array is of length {\tt nentries}.

NOTE: For better efficiency, use \Ref{HYPRE_StructMatrixSetBoxValues} to set coefficients a box at a time.

2.8.3.29 HYPRE_StructStencilCreate()

```
HYPRE_Int HYPRE_StructStencilCreate (
            HYPRE_Int ndim,
            HYPRE_Int size,
            HYPRE_StructStencil * stencil )
```

Create a stencil object for the specified number of spatial dimensions and stencil entries.

2.8.3.30 HYPRE_StructStencilDestroy()

```
HYPRE_Int HYPRE_StructStencilDestroy (
            HYPRE_StructStencil stencil )
```

Destroy a stencil object.

2.8.3.31 HYPRE_StructStencilSetElement()

```
HYPRE_Int HYPRE_StructStencilSetElement (
            HYPRE_StructStencil stencil,
            HYPRE_Int entry,
            HYPRE_Int * offset )
```

Set a stencil entry.

NOTE: The name of this routine will eventually be changed to {\tt HYPRE_StructStencilSetEntry}.

2.8.3.32 HYPRE_StructVectorAddToBoxValues()

```
HYPRE_Int HYPRE_StructVectorAddToBoxValues (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Complex * values )
```

Add to vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructVectorSetBox↩ Values}.

2.8.3.33 HYPRE_StructVectorAddToBoxValues2()

```
HYPRE_Int HYPRE_StructVectorAddToBoxValues2 (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Add to vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructVectorSetBox↩ Values2}.

2.8.3.34 HYPRE_StructVectorAddToValues()

```
HYPRE_Int HYPRE_StructVectorAddToValues (
            HYPRE_StructVector vector,
            HYPRE_Int * index,
            HYPRE_Complex value )
```

Add to vector coefficients index by index.

NOTE: For better efficiency, use \Ref{HYPRE_StructVectorAddToBoxValues} to set coefficients a box at a time.

2.8.3.35 HYPRE_StructVectorAssemble()

```
HYPRE_Int HYPRE_StructVectorAssemble (
            HYPRE_StructVector vector )
```

Finalize the construction of the vector before using.

2.8.3.36 HYPRE_StructVectorCreate()

```
HYPRE_Int HYPRE_StructVectorCreate (
            MPI_Comm comm,
            HYPRE_StructGrid grid,
            HYPRE_StructVector * vector )
```

The vector object. Create a vector object.

2.8.3.37 HYPRE_StructVectorDestroy()

```
HYPRE_Int HYPRE_StructVectorDestroy (
            HYPRE_StructVector vector )
```

Destroy a vector object.

2.8.3.38 HYPRE_StructVectorGetBoxValues()

```
HYPRE_Int HYPRE_StructVectorGetBoxValues (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Complex * values )
```

Get vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructVectorSetBoxValues}.

2.8.3.39 HYPRE_StructVectorGetBoxValues2()

```
HYPRE_Int HYPRE_StructVectorGetBoxValues2 (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Get vector coefficients a box at a time. The data in {\tt values} is ordered as in \Ref{HYPRE_StructVectorSetBox↵ Values2}.

2.8.3.40 HYPRE_StructVectorGetMigrateCommPkg()

```
HYPRE_Int HYPRE_StructVectorGetMigrateCommPkg (
            HYPRE_StructVector from_vector,
            HYPRE_StructVector to_vector,
            HYPRE_CommPkg * comm_pkg )
```

2.8.3.41 HYPRE_StructVectorGetValues()

```
HYPRE_Int HYPRE_StructVectorGetValues (
            HYPRE_StructVector vector,
            HYPRE_Int * index,
            HYPRE_Complex * value )
```

Get vector coefficients index by index.

NOTE: For better efficiency, use \Ref{HYPRE_StructVectorGetBoxValues} to get coefficients a box at a time.

2.8.3.42 HYPRE_StructVectorInitialize()

```
HYPRE_Int HYPRE_StructVectorInitialize (
            HYPRE_StructVector vector )
```

Prepare a vector object for setting coefficient values.

2.8.3.43 HYPRE_StructVectorMigrate()

```
HYPRE_Int HYPRE_StructVectorMigrate (
            HYPRE_CommPkg comm_pkg,
            HYPRE_StructVector from_vector,
            HYPRE_StructVector to_vector )
```

2.8.3.44 HYPRE_StructVectorPrint()

```
HYPRE_Int HYPRE_StructVectorPrint (
            const char * filename,
            HYPRE_StructVector vector,
            HYPRE_Int all )
```

Print the vector to file. This is mainly for debugging purposes.

2.8.3.45 HYPRE_StructVectorSetBoxValues()

```
HYPRE_Int HYPRE_StructVectorSetBoxValues (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Complex * values )
```

Set vector coefficients a box at a time. The data in {\tt values} is ordered as follows:

\begin{verbatim} m = 0; for (k = ilower[2]; k <= iupper[2]; k++) for (j = ilower[1]; j <= iupper[1]; j++) for (i = ilower[0]; i <= iupper[0]; i++) { values[m] = ...; m++; } \end{verbatim}

2.8.3.46 HYPRE_StructVectorSetBoxValues2()

```
HYPRE_Int HYPRE_StructVectorSetBoxValues2 (
            HYPRE_StructVector vector,
            HYPRE_Int * ilower,
            HYPRE_Int * iupper,
            HYPRE_Int * vilower,
            HYPRE_Int * viupper,
            HYPRE_Complex * values )
```

Set vector coefficients a box at a time. The {\tt values} array is logically box shaped with value-box extents {\tt vilower} and {\tt viupper} that must contain the set-box extents {\tt ilower} and {\tt iupper} . The data in the {\tt values} array is ordered as in \Ref{HYPRE_StructVectorSetBoxValues}, but based on the value-box extents.

2.8.3.47 HYPRE_StructVectorSetConstantValues()

```
HYPRE_Int HYPRE_StructVectorSetConstantValues (
            HYPRE_StructVector vector,
            HYPRE_Complex values )
```

2.8.3.48 HYPRE_StructVectorSetNumGhost()

```
HYPRE_Int HYPRE_StructVectorSetNumGhost (
            HYPRE_StructVector vector,
            HYPRE_Int * num_ghost )
```

2.8.3.49 HYPRE_StructVectorSetValues()

```
HYPRE_Int HYPRE_StructVectorSetValues (
            HYPRE_StructVector vector,
            HYPRE_Int * index,
            HYPRE_Complex value )
```

Set vector coefficients index by index.

NOTE: For better efficiency, use \Ref{HYPRE_StructVectorSetBoxValues} to set coefficients a box at a time.