

hypre **Reference Manual**

— Version 1.9.0b —

Contents

1	Matrix and Vector Building Interfaces (Conceptual Interfaces) —	3
1.1	IJ Matrix Builder —	3
1.2	IJ Vector Builder —	7
2	Operator Interface —	12
3	Vector Interface —	14
4	Matrices and Vectors —	16
4.1	IJParCSR Matrix —	16
4.2	IJParCSR Vector —	24
5	Solver Interface —	31
6	ParCSR Solvers — <i>Linear solvers for sparse matrix systems</i>	33
6.1	ParCSRDiagScale Solver —	33
6.2	ParCSR BoomerAMG Solver —	37
7	PreconditionedSolver Interface —	42
8	Preconditioned Solvers —	43
8.1	PCG Preconditioned Solver —	43
8.2	GMRES Preconditioned Solver —	46
	Class Graph	50

Matrix and Vector Building Interfaces (Conceptual Interfaces)

Names

1.1	IJ Matrix Builder	3
1.2	IJ Vector Builder	7

1.1

IJ Matrix Builder

Names

1.1.1	struct bHYPRE_IJBuildMatrix__object <i>Symbol "bHYPRE"</i>	4
1.1.2	int32_t bHYPRE_IJBuildMatrix_SetLocalRange (bHYPRE_IJBuildMatrix self, int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper) <i>Set the local range for a matrix object</i>	5
1.1.3	int32_t bHYPRE_IJBuildMatrix_SetValues (bHYPRE_IJBuildMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array* values) <i>Sets values for nrows of the matrix</i>	5
1.1.4	int32_t bHYPRE_IJBuildMatrix_AddToValues (bHYPRE_IJBuildMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array* values) <i>Adds to values for nrows of the matrix</i>	6
	int32_t	

	bHYPRE_IJBuildMatrix_GetLocalRange (bHYPRE_IJBuildMatrix self, int32_t* ilower, int32_t* iupper, int32_t* jlower, int32_t* jupper) <i>Gets range of rows owned by this processor and range of column partitioning for this processor</i>	
	int32_t bHYPRE_IJBuildMatrix_GetRowCounts (bHYPRE_IJBuildMatrix self, int32_t nrows, struct SIDL_int_array* rows, struct SIDL_int_array** ncols) <i>Gets number of nonzeros elements for nrows rows specified in rows and returns them in ncols, which needs to be allocated by the user</i>	
1.1.5	int32_t bHYPRE_IJBuildMatrix_GetValues (bHYPRE_IJBuildMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array** values) <i>Gets values for nrows rows or partial rows of the matrix</i>	6
1.1.6	int32_t bHYPRE_IJBuildMatrix_SetRowSizes (bHYPRE_IJBuildMatrix self, struct SIDL_int_array* sizes) <i>(Optional) Set the max number of nonzeros to expect in each row</i>	6
1.1.7	int32_t bHYPRE_IJBuildMatrix_Print (bHYPRE_IJBuildMatrix self, const char* filename) <i>Print the matrix to file</i>	7
1.1.8	int32_t bHYPRE_IJBuildMatrix_Read (bHYPRE_IJBuildMatrix self, const char* filename, void* comm) <i>Read the matrix from file</i>	7
	bHYPRE_IJBuildMatrix bHYPRE_IJBuildMatrix__cast (void* obj) <i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_IJBuildMatrix__cast2 (void* obj, const char* type) <i>String cast method for interface and class type conversions</i>	

1.1.1

struct **bHYPRE_IJBuildMatrix__object**

This interface represents a linear-algebraic conceptual view of a linear system. The 'I' and 'J' in the name are meant to be mnemonic for the traditional matrix notation $A(I,J)$.

1.1.2

```
int32_t  
bHYPRE_IJBuildMatrix_SetLocalRange ( bHYPRE_IJBuildMatrix self,  
int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper)
```

Set the local range for a matrix object. Each process owns some unique consecutive range of rows, indicated by the global row indices `ilower` and `iupper`. The row data is required to be such that the value of `ilower` on any process p be exactly one more than the value of `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, `jlower` and `jupper` typically should match `ilower` and `iupper`, respectively. For rectangular matrices, `jlower` and `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 `jlower` and `jupper` to store the diagonal blocks (rectangular in general) of the matrix separately from the rest of the matrix.

Collective.

1.1.3

```
int32_t  
bHYPRE_IJBuildMatrix_SetValues ( bHYPRE_IJBuildMatrix self, int32_t  
nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct  
SIDL_int_array* cols, struct SIDL_double_array* values)
```

Sets values for `nrows` of the matrix. The arrays `ncols` and `rows` are of dimension `nrows` and contain the number of columns in each row and the row indices, respectively. The array `cols` contains the column indices for each of the `rows`, and is ordered by rows. The data in the `values` array corresponds directly to the column entries in `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.

Not collective.

1.1.4

```
int32_t  
bHYPRE_IJBuildMatrix_AddToValues ( bHYPRE_IJBuildMatrix self, int32_t  
nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct  
SIDL_int_array* cols, struct SIDL_double_array* values)
```

Adds to values for **nrows** of the matrix. Usage details are analogous to **SetValues**. Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one.

Not collective.

1.1.5

```
int32_t  
bHYPRE_IJBuildMatrix_GetValues ( bHYPRE_IJBuildMatrix self, int32_t  
nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct  
SIDL_int_array* cols, struct SIDL_double_array** values)
```

Gets values for **nrows** rows or partial rows of the matrix. Usage details are analogous to **SetValues**.

1.1.6

```
int32_t  
bHYPRE_IJBuildMatrix_SetRowSizes ( bHYPRE_IJBuildMatrix self, struct  
SIDL_int_array* sizes)
```

(Optional) Set the max number of nonzeros to expect in each row. The array **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.

1.1.7

```
int32_t
bHYPRE_IJBuildMatrix_Print ( bHYPRE_IJBuildMatrix self, const char*
filename)
```

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

1.1.8

```
int32_t
bHYPRE_IJBuildMatrix_Read ( bHYPRE_IJBuildMatrix self, const char*
filename, void* comm)
```

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

1.2

IJ Vector Builder

Names

1.2.1	struct bHYPRE_IJBuildVector__object <i>Symbol "bHYPRE"</i>	8
1.2.2	int32_t	

	bHYPRE_IJBuildVector_SetLocalRange (bHYPRE_IJBuildVector self, int32_t jlower, int32_t jupper) <i>Set the local range for a vector object</i>	9
1.2.3	int32_t bHYPRE_IJBuildVector_SetValues (bHYPRE_IJBuildVector self, int32_t nvalues, struct SIDL_int_array* indices, struct SIDL_double_array* values) <i>Sets values in vector</i>	9
1.2.4	int32_t bHYPRE_IJBuildVector_AddToValues (bHYPRE_IJBuildVector self, int32_t nvalues, struct SIDL_int_array* indices, struct SIDL_double_array* values) <i>Adds to values in vector</i>	9
	int32_t bHYPRE_IJBuildVector_GetLocalRange (bHYPRE_IJBuildVector self, int32_t* jlower, int32_t* jupper) <i>Returns range of the part of the vector owned by this processor</i>	
1.2.5	int32_t bHYPRE_IJBuildVector_GetValues (bHYPRE_IJBuildVector self, int32_t nvalues, struct SIDL_int_array* indices, struct SIDL_double_array** values) <i>Gets values in vector</i>	10
1.2.6	int32_t bHYPRE_IJBuildVector_Print (bHYPRE_IJBuildVector self, const char* filename) <i>Print the vector to file</i>	10
1.2.7	int32_t bHYPRE_IJBuildVector_Read (bHYPRE_IJBuildVector self, const char* filename, void* comm) <i>Read the vector from file</i>	10
	bHYPRE_IJBuildVector bHYPRE_IJBuildVector__cast (void* obj) <i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_IJBuildVector__cast2 (void* obj, const char* type) <i>String cast method for interface and class type conversions</i>	

1.2.1

```
struct bHYPRE_IJBuildVector_object
```


1.2.2

```
int32_t  
bHYPRE_IJBuildVector_SetLocalRange ( bHYPRE_IJBuildVector self,  
int32_t jlower, int32_t jupper)
```

Set the local range for a vector object. Each process owns some unique consecutive range of vector unknowns, indicated by the global indices `jlower` and `jupper`. The data is required to be such that the value of `jlower` on any process p be exactly one more than the value of `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.

1.2.3

```
int32_t  
bHYPRE_IJBuildVector_SetValues ( bHYPRE_IJBuildVector self, int32_t  
nvalues, struct SIDL_int__array* indices, struct SIDL_double__array* values)
```

Sets values in vector. The arrays `values` and `indices` are of dimension `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.

Not collective.

1.2.4

```
int32_t  
bHYPRE_IJBuildVector_AddToValues ( bHYPRE_IJBuildVector self, int32_t  
nvalues, struct SIDL_int__array* indices, struct SIDL_double__array* values)
```

Adds to values in vector. Usage details are analogous to `SetValues`.

Not collective.

1.2.5

```
int32_t
bHYPRE_IJBuildVector_GetValues ( bHYPRE_IJBuildVector self, int32_t
nvalues, struct SIDL_int__array* indices, struct SIDL_double__array** values)
```

Gets values in vector. Usage details are analogous to `SetValues`.

Not collective.

1.2.6

```
int32_t
bHYPRE_IJBuildVector_Print ( bHYPRE_IJBuildVector self, const char*
filename)
```

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

1.2.7

```
int32_t
bHYPRE_IJBuildVector_Read ( bHYPRE_IJBuildVector self, const char*
filename, void* comm)
```

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

Operator Interface

Names

2.1	struct bHYPRE_Operator__object <i>Symbol "bHYPRE"</i>	13
int32_t	bHYPRE_Operator_SetCommunicator (bHYPRE_Operator self, void* mpi_comm) <i>Set the MPI Communicator</i>	
int32_t	bHYPRE_Operator_SetIntParameter (bHYPRE_Operator self, const char* name, int32_t value) <i>Set the int parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetDoubleParameter (bHYPRE_Operator self, const char* name, double value) <i>Set the double parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetStringParameter (bHYPRE_Operator self, const char* name, const char* value) <i>Set the string parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetIntArray1Parameter (bHYPRE_Operator self, const char* name, struct SIDL_int_array* value) <i>Set the int 1-D array parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetIntArray2Parameter (bHYPRE_Operator self, const char* name, struct SIDL_int_array* value) <i>Set the int 2-D array parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetDoubleArray1Parameter (bHYPRE_Operator self, const char* name, struct SIDL_double_array* value) <i>Set the double 1-D array parameter associated with name</i>	
int32_t	bHYPRE_Operator_SetDoubleArray2Parameter (bHYPRE_Operator self, const char* name, struct SIDL_double_array* value) <i>Set the double 2-D array parameter associated with name</i>	
int32_t		

bHYPRE_Operator_GetIntValue (bHYPRE_Operator self,
const char* name, int32_t* value)
Set the int parameter associated with name

int32_t
bHYPRE_Operator_GetDoubleValue (bHYPRE_Operator self,
const char* name, double* value)
Get the double parameter associated with name

int32_t
bHYPRE_Operator_Setup (bHYPRE_Operator self, bHYPRE_Vector b,
bHYPRE_Vector x)
*(Optional) Do any preprocessing that may be necessary in order to execute
Apply*

int32_t
bHYPRE_Operator_Apply (bHYPRE_Operator self, bHYPRE_Vector b,
bHYPRE_Vector* x)
Apply the operator to b, returning x

bHYPRE_Operator
bHYPRE_Operator__cast (void* obj)
Cast method for interface and class type conversions

void*
bHYPRE_Operator__cast2 (void* obj, const char* type)
String cast method for interface and class type conversions

2.1

```
struct bHYPRE_Operator__object
```

Symbol "bHYPRE.Operator" (version 1.0.0)

An Operator is anything that maps one Vector to another. The terms **Setup** and **Apply** are reserved for Operators. The implementation is allowed to assume that supplied parameter arrays will not be destroyed.

Vector Interface

Names

3.1	struct bHYPRE_Vector__object <i>Symbol "bHYPRE"</i>	14
	int32_t bHYPRE_Vector_Clear (bHYPRE_Vector self) <i>Set self to 0</i>	
	int32_t bHYPRE_Vector_Copy (bHYPRE_Vector self, bHYPRE_Vector x) <i>Copy x into self</i>	
3.2	int32_t bHYPRE_Vector_Clone (bHYPRE_Vector self, bHYPRE_Vector* x) <i>Create an x compatible with self</i>	15
	int32_t bHYPRE_Vector_Scale (bHYPRE_Vector self, double a) <i>Scale self by a</i>	
	int32_t bHYPRE_Vector_Dot (bHYPRE_Vector self, bHYPRE_Vector x, double* d) <i>Compute d, the inner-product of self and x</i>	
	int32_t bHYPRE_Vector_Axpy (bHYPRE_Vector self, double a, bHYPRE_Vector x) <i>Add a*x to self</i>	
	bHYPRE_Vector bHYPRE_Vector__cast (void* obj) <i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_Vector__cast2 (void* obj, const char* type) <i>String cast method for interface and class type conversions</i>	

3.1

```
struct bHYPRE_Vector__object
```

Symbol "bHYPRE.Vector" (version 1.0.0)

```
int32_t bHYPRE_Vector_Clone ( bHYPRE_Vector self, bHYPRE_Vector* x)
```

Create an `x` compatible with `self`.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

Matrices and Vectors

Names

4.1	IJParCSR Matrix	
	16
4.2	IJParCSR Vector	
	24

4.1

IJParCSR Matrix

Names

4.1.1	struct bHYPRE_IJParCSRMatrix__object <i>Symbol "bHYPRE"</i>	20
	extern C bHYPRE_IJParCSRMatrix bHYPRE_IJParCSRMatrix__create (void) <i>Constructor function for the class</i>	
4.1.2	int32_t bHYPRE_IJParCSRMatrix_SetDiagOffdSizes (bHYPRE_IJParCSRMatrix self, struct SIDL_int__array* diag_sizes, struct SIDL_int__array* offdiag_sizes) <i>(Optional) Set the max number of nonzeros to expect in each row of the diagonal and off-diagonal blocks</i>	20
	int32_t bHYPRE_IJParCSRMatrix_SetCommunicator (bHYPRE_IJParCSRMatrix self, void* mpi_comm) <i>Set the MPI Communicator</i>	
	int32_t bHYPRE_IJParCSRMatrix_Initialize (bHYPRE_IJParCSRMatrix self) <i>Prepare an object for setting coefficient values, whether for the first time or subsequently</i>	
4.1.3	int32_t bHYPRE_IJParCSRMatrix_Assemble (bHYPRE_IJParCSRMatrix self) <i>Finalize the construction of an object before using, either for the first time or on subsequent uses</i>	21
4.1.4	int32_t	

	bHYPRE_IJParCSRMatrix_GetObject (bHYPRE_IJParCSRMatrix self, SIDL_BaseInterface* A) <i>The problem definition interface is a builder that creates an object that contains the problem definition information, e</i>	21
4.1.5	int32_t bHYPRE_IJParCSRMatrix_SetLocalRange (bHYPRE_IJParCSRMatrix self, int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper) <i>Set the local range for a matrix object</i>	21
4.1.6	int32_t bHYPRE_IJParCSRMatrix_SetValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array* values) <i>Sets values for nrows of the matrix</i>	22
4.1.7	int32_t bHYPRE_IJParCSRMatrix_AddToValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array* values) <i>Adds to values for nrows of the matrix</i>	22
	int32_t bHYPRE_IJParCSRMatrix_GetLocalRange (bHYPRE_IJParCSRMatrix self, int32_t* ilower, int32_t* iupper, int32_t* jlower, int32_t* jupper) <i>Gets range of rows owned by this processor and range of column partitioning for this processor</i>	
	int32_t bHYPRE_IJParCSRMatrix_GetRowCounts (bHYPRE_IJParCSRMatrix self, int32_t nrows, struct SIDL_int_array* rows, struct SIDL_int_array** ncols) <i>Gets number of nonzeros elements for nrows rows specified in rows and returns them in ncols, which needs to be allocated by the user</i>	
4.1.8	int32_t	

	bHYPRE_IJParCSRMatrix_GetValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct SIDL_int_array* cols, struct SIDL_double_array** values)	
	<i>Gets values for nrows rows or partial rows of the matrix</i>	23
4.1.9	int32_t bHYPRE_IJParCSRMatrix_SetRowSizes (bHYPRE_IJParCSRMatrix self, struct SIDL_int_array* sizes)	
	<i>(Optional) Set the max number of nonzeros to expect in each row</i>	23
4.1.10	int32_t bHYPRE_IJParCSRMatrix_Print (bHYPRE_IJParCSRMatrix self, const char* filename)	
	<i>Print the matrix to file</i>	23
4.1.11	int32_t bHYPRE_IJParCSRMatrix_Read (bHYPRE_IJParCSRMatrix self, const char* filename, void* comm)	
	<i>Read the matrix from file</i>	24
	int32_t bHYPRE_IJParCSRMatrix_SetIntParameter (bHYPRE_IJParCSRMatrix self, const char* name, int32_t value)	
	<i>Set the int parameter associated with name</i>	
	int32_t bHYPRE_IJParCSRMatrix_SetDoubleParameter (bHYPRE_IJParCSRMatrix self, const char* name, double value)	
	<i>Set the double parameter associated with name</i>	
	int32_t bHYPRE_IJParCSRMatrix_SetStringParameter (bHYPRE_IJParCSRMatrix self, const char* name, const char* value)	
	<i>Set the string parameter associated with name</i>	
	int32_t bHYPRE_IJParCSRMatrix_SetIntArray1Parameter (bHYPRE_IJParCSRMatrix self, const char* name, struct SIDL_int_array* value)	
	<i>Set the int 1-D array parameter associated with name</i>	
	int32_t	

bHYPRE_IJParCSRMatrix_SetIntArray2Parameter (

bHYPRE_IJParCSRMatrix
self,
const char* name,
struct
SIDL_int__array*
value)

Set the int 2-D array parameter associated with name

int32_t

bHYPRE_IJParCSRMatrix_SetDoubleArray1Parameter (

bHYPRE_IJParCSRMatrix
self, const
char* name,
struct
SIDL_double__array*
value)

Set the double 1-D array parameter associated with name

int32_t

bHYPRE_IJParCSRMatrix_SetDoubleArray2Parameter (

bHYPRE_IJParCSRMatrix
self, const
char* name,
struct
SIDL_double__array*
value)

Set the double 2-D array parameter associated with name

int32_t

bHYPRE_IJParCSRMatrix_GetIntValue (bHYPRE_IJParCSRMatrix self,
const char* name, int32_t* value)

Set the int parameter associated with name

int32_t

bHYPRE_IJParCSRMatrix_GetDoubleValue (bHYPRE_IJParCSRMatrix
self, const char* name,
double* value)

Get the double parameter associated with name

int32_t

bHYPRE_IJParCSRMatrix_Setup (bHYPRE_IJParCSRMatrix self,
bHYPRE_Vector b, bHYPRE_Vector x)
(Optional) Do any preprocessing that may be necessary in order to execute
Apply

int32_t

bHYPRE_IJParCSRMatrix_Apply (bHYPRE_IJParCSRMatrix self,
bHYPRE_Vector b,
bHYPRE_Vector* x)

Apply the operator to b, returning x

4.1.12

int32_t

```

bHYPRE_IJParCSRMatrix_GetRow ( bHYPRE_IJParCSRMatrix self,
                                int32_t row, int32_t* size,
                                struct SIDL_int__array** col_ind,
                                struct SIDL_double__array** values)
    The GetRow method will allocate space for its two output arrays on the first
    call .....

```

24

```

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__cast ( void* obj)
    Cast method for interface and class type conversions

void*
bHYPRE_IJParCSRMatrix__cast2 ( void* obj, const char* type)
    String cast method for interface and class type conversions

```

4.1.1

```

struct bHYPRE_IJParCSRMatrix__object

```

Symbol "bHYPRE.IJParCSRMatrix" (version 1.0.0)

The IJParCSR matrix class.

Objects of this type can be cast to IJBuildMatrix, Operator, or CoefficientAccess objects using the `__cast` methods.

4.1.2

```

int32_t
bHYPRE_IJParCSRMatrix_SetDiagOffdSizes ( bHYPRE_IJParCSRMatrix
self, struct SIDL_int__array* diag_sizes, struct SIDL_int__array* offdiag_sizes)

```

(Optional) Set the max number of nonzeros to expect 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 `diag_sizes` and `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.

4.1.3

```
int32_t  
bHYPRE_IJParCSRMatrix_Assemble ( bHYPRE_IJParCSRMatrix self)
```

Finalize the construction of an object before using, either for the first time or on subsequent uses. **Initialize** and **Assemble** always appear in a matched set, with **Initialize** preceding **Assemble**. Values can only be set in between a call to **Initialize** and **Assemble**.

4.1.4

```
int32_t  
bHYPRE_IJParCSRMatrix_GetObject ( bHYPRE_IJParCSRMatrix self,  
SIDL_BaseInterface* A)
```

The problem definition interface is a *builder* that creates an object that contains the problem definition information, e.g. a matrix. To perform subsequent operations with that object, it must be returned from the problem definition object. **GetObject** performs this function. At compile time, the type of the returned object is unknown. Thus, the returned type is a `SIDL.BaseInterface`. `QueryInterface` or `Cast` must be used on the returned object to convert it into a known type.

4.1.5

```
int32_t  
bHYPRE_IJParCSRMatrix_SetLocalRange ( bHYPRE_IJParCSRMatrix self,  
int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper)
```

Set the local range for a matrix object. Each process owns some unique consecutive range of rows, indicated by the global row indices **ilower** and **iupper**. The row data is required to be such that the value of **ilower**

on any process p be exactly one more than the value of `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, `jlower` and `jupper` typically should match `ilower` and `iupper`, respectively. For rectangular matrices, `jlower` and `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 `jlower` and `jupper` to store the diagonal blocks (rectangular in general) of the matrix separately from the rest of the matrix.

Collective.

4.1.6

```
int32_t
bHYPRE_IJParCSRMatrix_SetValues ( bHYPRE_IJParCSRMatrix self,
int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct
SIDL_int_array* cols, struct SIDL_double_array* values)
```

Sets values for `nrows` of the matrix. The arrays `ncols` and `rows` are of dimension `nrows` and contain the number of columns in each row and the row indices, respectively. The array `cols` contains the column indices for each of the `rows`, and is ordered by rows. The data in the `values` array corresponds directly to the column entries in `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.

Not collective.

4.1.7

```
int32_t
bHYPRE_IJParCSRMatrix_AddToValues ( bHYPRE_IJParCSRMatrix self,
int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct
SIDL_int_array* cols, struct SIDL_double_array* values)
```

Adds to values for `nrows` of the matrix. Usage details are analogous to `SetValues`. Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one.

Not collective.

4.1.8

```
int32_t  
bHYPRE_IJParCSRMatrix_GetValues ( bHYPRE_IJParCSRMatrix self,  
int32_t nrows, struct SIDL_int_array* ncols, struct SIDL_int_array* rows, struct  
SIDL_int_array* cols, struct SIDL_double_array** values)
```

Gets values for **nrows** rows or partial rows of the matrix. Usage details are analogous to **SetValues**.

4.1.9

```
int32_t  
bHYPRE_IJParCSRMatrix_SetRowSizes ( bHYPRE_IJParCSRMatrix self,  
struct SIDL_int_array* sizes)
```

(Optional) Set the max number of nonzeros to expect in each row. The array **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.

4.1.10

```
int32_t  
bHYPRE_IJParCSRMatrix_Print ( bHYPRE_IJParCSRMatrix self, const  
char* filename)
```

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

4.1.11

```
int32_t
bHYPRE_IJParCSRMatrix_Read ( bHYPRE_IJParCSRMatrix self, const
char* filename, void* comm)
```

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

4.1.12

```
int32_t
bHYPRE_IJParCSRMatrix_GetRow ( bHYPRE_IJParCSRMatrix self, int32_t
row, int32_t* size, struct SIDL_int__array** col_ind, struct SIDL_double__array**
values)
```

The GetRow method will allocate space for its two output arrays on the first call. The space will be reused on subsequent calls. Thus the user must not delete them, yet must not depend on the data from GetRow to persist beyond the next GetRow call.

4.2

IJParCSR Vector

Names

4.2.1	struct bHYPRE_IJParCSRVector__object	
	<i>Symbol "bHYPRE"</i>	26
	extern C bHYPRE_IJParCSRVector	

	bHYPRE_IJParCSRVector__create (void) <i>Constructor function for the class</i>	
	int32_t bHYPRE_IJParCSRVector_Clear (bHYPRE_IJParCSRVector self) <i>Set self to 0</i>	
	int32_t bHYPRE_IJParCSRVector_Copy (bHYPRE_IJParCSRVector self, bHYPRE_Vector x) <i>Copy x into self</i>	
4.2.2	int32_t bHYPRE_IJParCSRVector_Clone (bHYPRE_IJParCSRVector self, bHYPRE_Vector* x) <i>Create an x compatible with self</i>	27
	int32_t bHYPRE_IJParCSRVector_Scale (bHYPRE_IJParCSRVector self, double a) <i>Scale self by a</i>	
	int32_t bHYPRE_IJParCSRVector_Dot (bHYPRE_IJParCSRVector self, bHYPRE_Vector x, double* d) <i>Compute d, the inner-product of self and x</i>	
	int32_t bHYPRE_IJParCSRVector_Axpy (bHYPRE_IJParCSRVector self, double a, bHYPRE_Vector x) <i>Add a*x to self</i>	
	int32_t bHYPRE_IJParCSRVector_SetCommunicator (bHYPRE_IJParCSRVector self, void* mpi_comm) <i>Set the MPI Communicator</i>	
	int32_t bHYPRE_IJParCSRVector_Initialize (bHYPRE_IJParCSRVector self) <i>Prepare an object for setting coefficient values, whether for the first time or subsequently</i>	
4.2.3	int32_t bHYPRE_IJParCSRVector_Assemble (bHYPRE_IJParCSRVector self) <i>Finalize the construction of an object before using, either for the first time or on subsequent uses</i>	27
4.2.4	int32_t bHYPRE_IJParCSRVector_GetObject (bHYPRE_IJParCSRVector self, SIDL_BaseInterface* A) <i>The problem definition interface is a builder that creates an object that con- tains the problem definition information, e</i>	27
4.2.5	int32_t bHYPRE_IJParCSRVector_SetLocalRange (bHYPRE_IJParCSRVector self, int32_t jlower, int32_t jupper) <i>Set the local range for a vector object</i>	28
4.2.6	int32_t	

	bHYPRE_IJParCSRVector_SetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, struct SIDL_int__array* indices, struct SIDL_double__array* values)	
	<i>Sets values in vector</i>	28
4.2.7	int32_t bHYPRE_IJParCSRVector_AddToValues (bHYPRE_IJParCSRVector self, int32_t nvalues, struct SIDL_int__array* indices, struct SIDL_double__array* values)	
	<i>Adds to values in vector</i>	29
	int32_t bHYPRE_IJParCSRVector_GetLocalRange (bHYPRE_IJParCSRVector self, int32_t* jlower, int32_t* jupper)	
	<i>Returns range of the part of the vector owned by this processor</i>	
4.2.8	int32_t bHYPRE_IJParCSRVector_GetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, struct SIDL_int__array* indices, struct SIDL_double__array** values)	
	<i>Gets values in vector</i>	29
4.2.9	int32_t bHYPRE_IJParCSRVector_Print (bHYPRE_IJParCSRVector self, const char* filename)	
	<i>Print the vector to file</i>	29
4.2.10	int32_t bHYPRE_IJParCSRVector_Read (bHYPRE_IJParCSRVector self, const char* filename, void* comm)	
	<i>Read the vector from file</i>	30
	bHYPRE_IJParCSRVector bHYPRE_IJParCSRVector__cast (void* obj)	
	<i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_IJParCSRVector__cast2 (void* obj, const char* type)	
	<i>String cast method for interface and class type conversions</i>	

4.2.1

```
struct bHYPRE_IJParCSRVector__object
```

Symbol "bHYPRE_IJParCSRVector" (version 1.0.0)

The IJParCSR vector class.

Objects of this type can be cast to IJBuildVector or Vector objects using the `__cast` methods.

4.2.2

```
int32_t  
bHYPRE_IJParCSRVector_Clone ( bHYPRE_IJParCSRVector self,  
bHYPRE_Vector* x)
```

Create an `x` compatible with `self`.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

4.2.3

```
int32_t  
bHYPRE_IJParCSRVector_Assemble ( bHYPRE_IJParCSRVector self)
```

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with `Initialize` preceding `Assemble`. Values can only be set in between a call to `Initialize` and `Assemble`.

4.2.4

```
int32_t  
bHYPRE_IJParCSRVector_GetObject ( bHYPRE_IJParCSRVector self,  
SIDL_BaseInterface* A)
```

The problem definition interface is a *builder* that creates an object that contains the problem definition information, e.g. a matrix. To perform subsequent operations with that object, it must be returned from the problem definition object. `GetObject` performs this function. At compile time, the type of the returned object is unknown. Thus, the returned type is a `SIDL.BaseInterface`. `QueryInterface` or `Cast` must be used on the returned object to convert it into a known type.

4.2.5

```
int32_t  
bHYPRE_IJParCSRVector_SetLocalRange ( bHYPRE_IJParCSRVector self,  
int32_t jlower, int32_t jupper)
```

Set the local range for a vector object. Each process owns some unique consecutive range of vector unknowns, indicated by the global indices `jlower` and `jupper`. The data is required to be such that the value of `jlower` on any process p be exactly one more than the value of `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.

4.2.6

```
int32_t  
bHYPRE_IJParCSRVector_SetValues ( bHYPRE_IJParCSRVector self, int32_t  
nvalues, struct SIDL_int__array* indices, struct SIDL_double__array* values)
```

Sets values in vector. The arrays `values` and `indices` are of dimension `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.

Not collective.

4.2.7

```
int32_t  
bHYPRE_IJParCSRVector_AddToValues ( bHYPRE_IJParCSRVector self,  
int32_t nvalues, struct SIDL_int_array* indices, struct SIDL_double_array* values)
```

Adds to values in vector. Usage details are analogous to **SetValues**.

Not collective.

4.2.8

```
int32_t  
bHYPRE_IJParCSRVector_GetValues ( bHYPRE_IJParCSRVector self,  
int32_t nvalues, struct SIDL_int_array* indices, struct SIDL_double_array** values)
```

Gets values in vector. Usage details are analogous to **SetValues**.

Not collective.

4.2.9

```
int32_t  
bHYPRE_IJParCSRVector_Print ( bHYPRE_IJParCSRVector self, const char*  
filename)
```

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

```
int32_t  
bHYPRE_IJParCSRVector_Read ( bHYPRE_IJParCSRVector self, const char*  
filename, void* comm)
```

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

5.1	<pre> struct bHYPRE_Solver__object Symbol "bHYPRE" int32_t bHYPRE_Solver_SetOperator (bHYPRE_Solver self, bHYPRE_Operator A) Set the operator for the linear system being solved int32_t bHYPRE_Solver_SetTolerance (bHYPRE_Solver self, double tolerance) (Optional) Set the convergence tolerance int32_t bHYPRE_Solver_SetMaxIterations (bHYPRE_Solver self, int32_t max_iterations) (Optional) Set maximum number of iterations </pre>	32
5.2	<pre> int32_t bHYPRE_Solver_SetLogging (bHYPRE_Solver self, int32_t level) (Optional) Set the logging level, specifying the degree of additional informa- tional data to be accumulated </pre>	32
5.3	<pre> int32_t bHYPRE_Solver_SetPrintLevel (bHYPRE_Solver self, int32_t level) (Optional) Set the print level, specifying the degree of informational data to be printed either to the screen or to a file int32_t bHYPRE_Solver_GetNumIterations (bHYPRE_Solver self, int32_t* num_iterations) (Optional) Return the number of iterations taken int32_t bHYPRE_Solver_GetRelResidualNorm (bHYPRE_Solver self, double* norm) (Optional) Return the norm of the relative residual bHYPRE_Solver bHYPRE_Solver__cast (void* obj) Cast method for interface and class type conversions void* bHYPRE_Solver__cast2 (void* obj, const char* type) String cast method for interface and class type conversions </pre>	32

5.1

```
struct bHYPRE_Solver__object
```

Symbol "bHYPRE.Solver" (version 1.0.0)

5.2

```
int32_t bHYPRE_Solver_SetLogging ( bHYPRE_Solver self, int32_t level)
```

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

5.3

```
int32_t bHYPRE_Solver_SetPrintLevel ( bHYPRE_Solver self, int32_t level)
```

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

ParCSR Solvers

Names

6.1	ParCSRDiagScale Solver	33
6.2	ParCSR BoomerAMG Solver	37

These solvers use matrix/vector storage schemes that are tailored for general sparse matrix systems.

6.1

ParCSRDiagScale Solver

Names

6.1.1	struct bHYPRE_ParCSRDiagScale_object <i>Symbol "bHYPRE"</i>	36
	extern C bHYPRE_ParCSRDiagScale bHYPRE_ParCSRDiagScale__create (void) <i>Constructor function for the class</i>	
	int32_t bHYPRE_ParCSRDiagScale_SetCommunicator (bHYPRE_ParCSRDiagScale self, void* mpi_comm) <i>Set the MPI Communicator</i>	
	int32_t bHYPRE_ParCSRDiagScale_SetIntParameter (bHYPRE_ParCSRDiagScale self, const char* name, int32_t value) <i>Set the int parameter associated with name</i>	
	int32_t	

bHYPRE_ParCSRDiagScale_SetDoubleParameter (
bHYPRE_ParCSRDiagScale
self, const char* name,
double value)

Set the double parameter associated with name

int32_t

bHYPRE_ParCSRDiagScale_SetStringParameter (
bHYPRE_ParCSRDiagScale
self, const char* name,
const char* value)

Set the string parameter associated with name

int32_t

bHYPRE_ParCSRDiagScale_SetIntArray1Parameter (
bHYPRE_ParCSRDiagScale
self,
const char* name,
struct
SIDL_int_array*
value)

Set the int 1-D array parameter associated with name

int32_t

bHYPRE_ParCSRDiagScale_SetIntArray2Parameter (
bHYPRE_ParCSRDiagScale
self,
const char* name,
struct
SIDL_int_array*
value)

Set the int 2-D array parameter associated with name

int32_t

bHYPRE_ParCSRDiagScale_SetDoubleArray1Parameter (
bHYPRE_ParCSRDiagScale
self, const
char* name,
struct
SIDL_double_array*
value)

Set the double 1-D array parameter associated with name

int32_t

bHYPRE_ParCSRDiagScale_SetDoubleArray2Parameter (
bHYPRE_ParCSRDiagScale
self, const
char* name,
struct
SIDL_double_array*
value)

Set the double 2-D array parameter associated with name

int32_t

	bHYPRE_ParCSRDiagScale_GetIntValue (bHYPRE_ParCSRDiagScale self, const char* name, int32_t* value) <i>Set the int parameter associated with name</i>	
	int32_t bHYPRE_ParCSRDiagScale_GetDoubleValue (bHYPRE_ParCSRDiagScale self, const char* name, double* value) <i>Get the double parameter associated with name</i>	
	int32_t bHYPRE_ParCSRDiagScale_Setup (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector x) <i>(Optional) Do any preprocessing that may be necessary in order to execute</i> Apply	
	int32_t bHYPRE_ParCSRDiagScale_Apply (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector* x) <i>Apply the operator to b, returning x</i>	
	int32_t bHYPRE_ParCSRDiagScale_SetOperator (bHYPRE_ParCSRDiagScale self, bHYPRE_Operator A) <i>Set the operator for the linear system being solved</i>	
	int32_t bHYPRE_ParCSRDiagScale_SetTolerance (bHYPRE_ParCSRDiagScale self, double tolerance) <i>(Optional) Set the convergence tolerance</i>	
	int32_t bHYPRE_ParCSRDiagScale_SetMaxIterations (bHYPRE_ParCSRDiagScale self, int32_t max_iterations) <i>(Optional) Set maximum number of iterations</i>	
6.1.2	int32_t bHYPRE_ParCSRDiagScale_SetLogging (bHYPRE_ParCSRDiagScale self, int32_t level) <i>(Optional) Set the logging level, specifying the degree of additional informa-</i> <i>tional data to be accumulated</i>	36
6.1.3	int32_t bHYPRE_ParCSRDiagScale_SetPrintLevel (bHYPRE_ParCSRDiagScale self, int32_t level) <i>(Optional) Set the print level, specifying the degree of informational data</i> <i>to be printed either to the screen or to a file</i>	37
	int32_t	

```

bHYPRE_ParCSRDiagScale_GetNumIterations (
    bHYPRE_ParCSRDiagScale
    self,
    int32_t* num_iterations)
    (Optional) Return the number of iterations taken

int32_t
bHYPRE_ParCSRDiagScale_GetRelResidualNorm (
    bHYPRE_ParCSRDiagScale
    self, double* norm)
    (Optional) Return the norm of the relative residual

bHYPRE_ParCSRDiagScale
bHYPRE_ParCSRDiagScale__cast ( void* obj)
    Cast method for interface and class type conversions

void*
bHYPRE_ParCSRDiagScale__cast2 ( void* obj, const char* type)
    String cast method for interface and class type conversions

```

6.1.1

```

struct bHYPRE_ParCSRDiagScale__object

```

Symbol "bHYPRE.ParCSRDiagScale" (version 1.0.0)

Diagonal scaling preconditioner for ParCSR matrix class.

Objects of this type can be cast to Solver objects using the `__cast` methods.

6.1.2

```

int32_t
bHYPRE_ParCSRDiagScale_SetLogging ( bHYPRE_ParCSRDiagScale self,
int32_t level)

```

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before Setup and Apply.

```
int32_t
bHYPRE_ParCSRDiagScale_SetPrintLevel ( bHYPRE_ParCSRDiagScale self,
int32_t level)
```

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

ParCSR BoomerAMG Solver

Names

```
6.2.1      struct bHYPRE_BoomerAMG_object
              Symbol "bHYPRE" ..... 39

extern C bHYPRE_BoomerAMG
bHYPRE_BoomerAMG_create (void)
              Constructor function for the class

int32_t
bHYPRE_BoomerAMG_SetCommunicator ( bHYPRE_BoomerAMG self,
                                   void* mpi_comm)
              Set the MPI Communicator

int32_t
bHYPRE_BoomerAMG_SetIntParameter ( bHYPRE_BoomerAMG self,
                                   const char* name, int32_t value)
              Set the int parameter associated with name

int32_t
bHYPRE_BoomerAMG_SetDoubleParameter ( bHYPRE_BoomerAMG
                                   self, const char* name,
                                   double value)
              Set the double parameter associated with name

int32_t
```

bHYPRE_BoomerAMG_SetStringParameter (bHYPRE_BoomerAMG self,
const char* name,
const char* value)

Set the string parameter associated with name

int32_t

bHYPRE_BoomerAMG_SetIntArray1Parameter (bHYPRE_BoomerAMG
self, const char* name,
struct SIDL_int__array*
value)

Set the int 1-D array parameter associated with name

int32_t

bHYPRE_BoomerAMG_SetIntArray2Parameter (bHYPRE_BoomerAMG
self, const char* name,
struct SIDL_int__array*
value)

Set the int 2-D array parameter associated with name

int32_t

bHYPRE_BoomerAMG_SetDoubleArray1Parameter (
bHYPRE_BoomerAMG
self,
const char* name,
struct
SIDL_double__array*
value)

Set the double 1-D array parameter associated with name

int32_t

bHYPRE_BoomerAMG_SetDoubleArray2Parameter (
bHYPRE_BoomerAMG
self,
const char* name,
struct
SIDL_double__array*
value)

Set the double 2-D array parameter associated with name

int32_t

bHYPRE_BoomerAMG_GetIntValue (bHYPRE_BoomerAMG self,
const char* name, int32_t* value)

Set the int parameter associated with name

int32_t

bHYPRE_BoomerAMG_GetDoubleValue (bHYPRE_BoomerAMG self,
const char* name,
double* value)

Get the double parameter associated with name

int32_t

bHYPRE_BoomerAMG_Setup (bHYPRE_BoomerAMG self,
bHYPRE_Vector b, bHYPRE_Vector x)
(Optional) Do any preprocessing that may be necessary in order to execute
Apply

int32_t

	bHYPRE_BoomerAMG_Apply (bHYPRE_BoomerAMG self, bHYPRE_Vector b, bHYPRE_Vector* x) <i>Apply the operator to b, returning x</i>	
	int32_t bHYPRE_BoomerAMG_SetOperator (bHYPRE_BoomerAMG self, bHYPRE_Operator A) <i>Set the operator for the linear system being solved</i>	
	int32_t bHYPRE_BoomerAMG_SetTolerance (bHYPRE_BoomerAMG self, double tolerance) <i>(Optional) Set the convergence tolerance</i>	
	int32_t bHYPRE_BoomerAMG_SetMaxIterations (bHYPRE_BoomerAMG self, int32_t max_iterations) <i>(Optional) Set maximum number of iterations</i>	
6.2.2	int32_t bHYPRE_BoomerAMG_SetLogging (bHYPRE_BoomerAMG self, int32_t level) <i>(Optional) Set the logging level, specifying the degree of additional informational data to be accumulated</i>	41
6.2.3	int32_t bHYPRE_BoomerAMG_SetPrintLevel (bHYPRE_BoomerAMG self, int32_t level) <i>(Optional) Set the print level, specifying the degree of informational data to be printed either to the screen or to a file</i>	41
	int32_t bHYPRE_BoomerAMG_GetNumIterations (bHYPRE_BoomerAMG self, int32_t* num_iterations) <i>(Optional) Return the number of iterations taken</i>	
	int32_t bHYPRE_BoomerAMG_GetRelResidualNorm (bHYPRE_BoomerAMG self, double* norm) <i>(Optional) Return the norm of the relative residual</i>	
	bHYPRE_BoomerAMG bHYPRE_BoomerAMG_cast (void* obj) <i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_BoomerAMG_cast2 (void* obj, const char* type) <i>String cast method for interface and class type conversions</i>	

6.2.1

```
struct bHYPRE_BoomerAMG_object
```

Symbol "bHYPRE.BoomerAMG" (version 1.0.0)

Algebraic multigrid solver, based on classical Ruge-Stueben.

The following optional parameters are available and may be set using the appropriate `Parameter` function (as indicated in parentheses):

MaxLevels (Int) - maximum number of multigrid levels.

StrongThreshold (Double) - AMG strength threshold.

MaxRowSum (Double) -

CoarsenType (Int) - type of parallel coarsening algorithm used.

MeasureType (Int) - type of measure used; local or global.

CycleType (Int) - type of cycle used; a V-cycle (default) or a W-cycle.

NumGridSweeps (IntArray 1D) - number of sweeps for fine and coarse grid, up and down cycle.

GridRelaxType (IntArray 1D) - type of smoother used on fine and coarse grid, up and down cycle.

GridRelaxPoints (IntArray 2D) - point ordering used in relaxation.

RelaxWeight (DoubleArray 1D) - relaxation weight for smoothed Jacobi and hybrid SOR.

TruncFactor (Double) - truncation factor for interpolation.

SmoothType (Int) - more complex smoothers.

SmoothNumLevels (Int) - number of levels for more complex smoothers.

SmoothNumSweeps (Int) - number of sweeps for more complex smoothers.

PrintFileName (String) - name of file printed to in association with `SetPrintLevel`. (not yet implemented).

NumFunctions (Int) - size of the system of PDEs (when using the systems version).

DOFFunc (IntArray 1D) - mapping that assigns the function to each variable (when using the systems version).

Variant (Int) - variant of Schwarz used.

Overlap (Int) - overlap for Schwarz.

DomainType (Int) - type of domain used for Schwarz.

SchwarzRlxWeight (Double) - the smoothing parameter for additive Schwarz.

DebugFlag (Int) -

Objects of this type can be cast to Solver objects using the `__cast` methods.


```
int32_t
bHYPRE_BoomerAMG_SetLogging ( bHYPRE_BoomerAMG self, int32_t
level)
```

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

```
int32_t
bHYPRE_BoomerAMG_SetPrintLevel ( bHYPRE_BoomerAMG self, int32_t
level)
```

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

PreconditionedSolver Interface

Names

7.1	struct bHYPRE_PreconditionedSolver_object <i>Symbol "bHYPRE"</i>	42
	int32_t bHYPRE_PreconditionedSolver_SetPreconditioner (bHYPRE_PreconditionedSolver self, bHYPRE_Solver s) <i>Set the preconditioner</i>	
	bHYPRE_PreconditionedSolver bHYPRE_PreconditionedSolver_cast (void* obj) <i>Cast method for interface and class type conversions</i>	
	void* bHYPRE_PreconditionedSolver_cast2 (void* obj, const char* type) <i>String cast method for interface and class type conversions</i>	

7.1

```
struct bHYPRE_PreconditionedSolver_object
```

Symbol "bHYPRE.PreconditionedSolver" (version 1.0.0)

Preconditioned Solvers

Names

8.1	PCG Preconditioned Solver	43
8.2	GMRES Preconditioned Solver	46

8.1

PCG Preconditioned Solver

Names

```

8.1.1      struct bHYPRE_PCG__object
              Symbol "bHYPRE" .....
extern C bHYPRE_PCG
bHYPRE_PCG_create (void)
              Constructor function for the class

int32_t
bHYPRE_PCG_SetCommunicator ( bHYPRE_PCG self, void* mpi_comm)
              Set the MPI Communicator

int32_t
bHYPRE_PCG_SetIntParameter ( bHYPRE_PCG self, const char* name,
                              int32_t value)
              Set the int parameter associated with name

int32_t
bHYPRE_PCG_SetDoubleParameter ( bHYPRE_PCG self,
                                  const char* name, double value)
              Set the double parameter associated with name

int32_t
bHYPRE_PCG_SetStringParameter ( bHYPRE_PCG self,
                                  const char* name, const char* value)
              Set the string parameter associated with name

int32_t
bHYPRE_PCG_SetIntArray1Parameter ( bHYPRE_PCG self,
                                      const char* name,
                                      struct SIDL_int_array* value)
              Set the int 1-D array parameter associated with name

int32_t

```

	bHYPRE_PCG_SetIntArray2Parameter (bHYPRE_PCG self, const char* name, struct SIDL_int__array* value) <i>Set the int 2-D array parameter associated with name</i>	
	int32_t bHYPRE_PCG_SetDoubleArray1Parameter (bHYPRE_PCG self, const char* name, struct SIDL_double_array* value) <i>Set the double 1-D array parameter associated with name</i>	
	int32_t bHYPRE_PCG_SetDoubleArray2Parameter (bHYPRE_PCG self, const char* name, struct SIDL_double_array* value) <i>Set the double 2-D array parameter associated with name</i>	
	int32_t bHYPRE_PCG_GetIntValue (bHYPRE_PCG self, const char* name, int32_t* value) <i>Set the int parameter associated with name</i>	
	int32_t bHYPRE_PCG_GetDoubleValue (bHYPRE_PCG self, const char* name, double* value) <i>Get the double parameter associated with name</i>	
	int32_t bHYPRE_PCG_Setup (bHYPRE_PCG self, bHYPRE_Vector b, bHYPRE_Vector x) <i>(Optional) Do any preprocessing that may be necessary in order to execute Apply</i>	
	int32_t bHYPRE_PCG_Apply (bHYPRE_PCG self, bHYPRE_Vector b, bHYPRE_Vector* x) <i>Apply the operator to b, returning x</i>	
	int32_t bHYPRE_PCG_SetOperator (bHYPRE_PCG self, bHYPRE_Operator A) <i>Set the operator for the linear system being solved</i>	
	int32_t bHYPRE_PCG_SetTolerance (bHYPRE_PCG self, double tolerance) <i>(Optional) Set the convergence tolerance</i>	
	int32_t bHYPRE_PCG_SetMaxIterations (bHYPRE_PCG self, int32_t max_iterations) <i>(Optional) Set maximum number of iterations</i>	
8.1.2	int32_t bHYPRE_PCG_SetLogging (bHYPRE_PCG self, int32_t level) <i>(Optional) Set the logging level, specifying the degree of additional informa- tional data to be accumulated</i>	45
8.1.3	int32_t	

bHYPRE_PCG_SetPrintLevel (bHYPRE_PCG self, int32_t level)
(Optional) Set the print level, specifying the degree of informational data to be printed either to the screen or to a file

int32_t
bHYPRE_PCG_GetNumIterations (bHYPRE_PCG self,
int32_t* num_iterations)
(Optional) Return the number of iterations taken

int32_t
bHYPRE_PCG_GetRelResidualNorm (bHYPRE_PCG self, double* norm)
(Optional) Return the norm of the relative residual

int32_t
bHYPRE_PCG_SetPreconditioner (bHYPRE_PCG self, bHYPRE_Solver s)
Set the preconditioner

bHYPRE_PCG
bHYPRE_PCG__cast (void* obj)
Cast method for interface and class type conversions

void*
bHYPRE_PCG__cast2 (void* obj, const char* type)
String cast method for interface and class type conversions

46

8.1.1

```
struct bHYPRE_PCG_object
```

Symbol "bHYPRE.PCG" (version 1.0.0)

Objects of this type can be cast to PreconditionedSolver objects using the `__cast` methods.

RDF: Documentation goes here.

8.1.2

```
int32_t bHYPRE_PCG_SetLogging ( bHYPRE_PCG self, int32_t level)
```

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated.

Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before Setup and Apply.

8.1.3

```
int32_t bHYPRE_PCG_SetPrintLevel ( bHYPRE_PCG self, int32_t level)
```

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before Setup and Apply.

8.2

GMRES Preconditioned Solver

Names

8.2.1	<pre> struct bHYPRE_GMRES_object <i>Symbol "bHYPRE"</i> extern C bHYPRE_GMRES bHYPRE_GMRES_create (void) <i>Constructor function for the class</i> int32_t bHYPRE_GMRES_SetCommunicator (bHYPRE_GMRES self, void* mpi_comm) <i>Set the MPI Communicator</i> int32_t bHYPRE_GMRES_SetIntParameter (bHYPRE_GMRES self, const char* name, int32_t value) <i>Set the int parameter associated with name</i> int32_t bHYPRE_GMRES_SetDoubleParameter (bHYPRE_GMRES self, const char* name, double value) <i>Set the double parameter associated with name</i> int32_t</pre>	48
-------	--	----

bHYPRE_GMRES_SetStringParameter (bHYPRE_GMRES self,
const char* name,
const char* value)
Set the string parameter associated with name

int32_t
bHYPRE_GMRES_SetIntArray1Parameter (bHYPRE_GMRES self,
const char* name,
struct SIDL_int__array* value)
Set the int 1-D array parameter associated with name

int32_t
bHYPRE_GMRES_SetIntArray2Parameter (bHYPRE_GMRES self,
const char* name,
struct SIDL_int__array* value)
Set the int 2-D array parameter associated with name

int32_t
bHYPRE_GMRES_SetDoubleArray1Parameter (bHYPRE_GMRES self,
const char* name, struct
SIDL_double__array*
value)
Set the double 1-D array parameter associated with name

int32_t
bHYPRE_GMRES_SetDoubleArray2Parameter (bHYPRE_GMRES self,
const char* name, struct
SIDL_double__array*
value)
Set the double 2-D array parameter associated with name

int32_t
bHYPRE_GMRES_GetIntValue (bHYPRE_GMRES self, const char* name,
int32_t* value)
Set the int parameter associated with name

int32_t
bHYPRE_GMRES_GetDoubleValue (bHYPRE_GMRES self,
const char* name, double* value)
Get the double parameter associated with name

int32_t
bHYPRE_GMRES_Setup (bHYPRE_GMRES self, bHYPRE_Vector b,
bHYPRE_Vector x)
*(Optional) Do any preprocessing that may be necessary in order to execute
Apply*

int32_t
bHYPRE_GMRES_Apply (bHYPRE_GMRES self, bHYPRE_Vector b,
bHYPRE_Vector* x)
Apply the operator to b, returning x

int32_t
bHYPRE_GMRES_SetOperator (bHYPRE_GMRES self,
bHYPRE_Operator A)
Set the operator for the linear system being solved

int32_t

	bHYPRE_GMRES_SetTolerance (bHYPRE_GMRES self, double tolerance) (Optional) Set the convergence tolerance	
	int32_t bHYPRE_GMRES_SetMaxIterations (bHYPRE_GMRES self, int32_t max_iterations) (Optional) Set maximum number of iterations	
8.2.2	int32_t bHYPRE_GMRES_SetLogging (bHYPRE_GMRES self, int32_t level) (Optional) Set the logging level, specifying the degree of additional informational data to be accumulated	49
8.2.3	int32_t bHYPRE_GMRES_SetPrintLevel (bHYPRE_GMRES self, int32_t level) (Optional) Set the print level, specifying the degree of informational data to be printed either to the screen or to a file	49
	int32_t bHYPRE_GMRES_GetNumIterations (bHYPRE_GMRES self, int32_t* num_iterations) (Optional) Return the number of iterations taken	
	int32_t bHYPRE_GMRES_GetRelResidualNorm (bHYPRE_GMRES self, double* norm) (Optional) Return the norm of the relative residual	
	int32_t bHYPRE_GMRES_SetPreconditioner (bHYPRE_GMRES self, bHYPRE_Solver s) Set the preconditioner	
	bHYPRE_GMRES bHYPRE_GMRES__cast (void* obj) Cast method for interface and class type conversions	
	void* bHYPRE_GMRES__cast2 (void* obj, const char* type) String cast method for interface and class type conversions	

8.2.1

```
struct bHYPRE_GMRES__object
```

Symbol "bHYPRE.GMRES" (version 1.0.0)

Objects of this type can be cast to PreconditionedSolver objects using the `__cast` methods.

RDF: Documentation goes here.

8.2.2

```
int32_t bHYPRE_GMRES_SetLogging ( bHYPRE_GMRES self, int32_t level)
```

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

8.2.3

```
int32_t  
bHYPRE_GMRES_SetPrintLevel ( bHYPRE_GMRES self, int32_t level)
```

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before **Setup** and **Apply**.

Class Graph