hypre Reference Manual

— Version 1.10.0b —

Contents

1 Matrix and Vector Building Interfaces (Conceptual Interfaces)

Names

1.1 IJ Matrix Builder

Names

1.1.1

struct **bHYPRE_IJMatrixView__object**

Symbol "bHYPRE.IJMatrixView" (version 1.0.0)

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_IJMatrixView_SetLocalRange (bHYPRE_IJMatrixView 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_IJMatrixView_SetValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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`. The last argument is the size of the cols and values arrays, i.e. the total number of nonzeros being provided, i.e. the sum of all values in ncols. This functin 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_IJMatrixView_AddToValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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_IJMatrixView_GetValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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

1.1.6

int32_t
bHYPRE_IJMatrixView_SetRowSizes (bHYPRE_IJMatrixView self, int32_t* sizes, int32_t nrows)

(Optional) Set the max number of nonzeros to expect in each row. The array `sizes` contains estimated sizes for each row on this process. The integer nrows is the number of rows in the local matrix. 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_IJMatrixView_Print (bHYPRE_IJMatrixView self, const char* filename)

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

1.1.8

int32_t
bHYPRE_IJMatrixView_Read (bHYPRE_IJMatrixView self, const char* filename, bHYPRE_MPICommunicator comm)

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

1.2

IJ Vector Builder

Names

1.2.1

struct **bHYPRE_IJVectorView__object**

Symbol "bHYPRE.IJVectorView" (version 1.0.0)

1.2.2

int32_t
bHYPRE_IJVectorView_SetLocalRange (bHYPRE_IJVectorView 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_IJVectorView_SetValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* 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_IJVectorView_AddToValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* values)

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

Not collective.

1.2.5

int32_t
bHYPRE_IJVectorView_GetValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* values)

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

Not collective.

1.2.6

int32_t
bHYPRE_IJVectorView_Print (bHYPRE_IJVectorView self, const char* filename)

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

1.2.7

int32_t
bHYPRE_IJVectorView_Read (bHYPRE_IJVectorView self, const char* filename, bHYPRE_MPICommunicator comm)

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

2

Operator Interface

Names

2.1 struct **bHYPRE_Operator__object**
Symbol "bHYPRE 14

extern C bHYPRE_Operator
bHYPRE_Operator__connect (const char *, sidl_BaseInterface *_ex)
RMI connector function for the class

2.2 int32_t
bHYPRE_Operator_SetCommunicator (bHYPRE_Operator self,
bHYPRE_MPICommunicator
mpi_comm)
Set the MPI Communicator 14

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

int32_t
bHYPRE_Operator_SetDoubleParameter (bHYPRE_Operator self,
const char* name, double value)
Set the double parameter associated with `name`

int32_t
bHYPRE_Operator_SetStringParameter (bHYPRE_Operator self,
const char* name,
const char* value)
Set the string parameter associated with `name`

int32_t
bHYPRE_Operator_SetIntArray1Parameter (bHYPRE_Operator self,
const char* name,
int32_t* value,
int32_t nvalues)
Set the int 1-D array parameter associated with `name`

int32_t
bHYPRE_Operator_SetIntArray2Parameter (bHYPRE_Operator self,
const char* name,
struct sidl_int__array* value)
Set the int 2-D array parameter associated with `name`

int32_t
bHYPRE_Operator_SetDoubleArray1Parameter (bHYPRE_Operator self,
const char* name,
double* value,
int32_t nvalues)
Set the double 1-D array parameter associated with `name`

int32_t

bHYPRE_Operator_SetDoubleArray2Parameter (bHYPRE_Operator self,
const char* name,
struct sidl_double__array*
value)
Set the double 2-D array parameter associated with `name`

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`

int32_t
bHYPRE_Operator_ApplyAdjoint (bHYPRE_Operator self,
bHYPRE_Vector b,
bHYPRE_Vector* x)
Apply the adjoint of the operator to `b`, *returning* `x`

struct bHYPRE_Operator__object* 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

void
bHYPRE_Operator__exec (bHYPRE_Operator self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_Operator__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_Operator__getURL (bHYPRE_Operator self)
Get the URL of the Implementation of this object (for RMI)

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.

2.2

int32_t
bHYPRE_Operator_SetCommunicator (bHYPRE_Operator self,
bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, use Create:

3

Vector Interface

Names

bHYPRE_Vector__getURL (bHYPRE_Vector self)
Get the URL of the Implementation of this object (for RMI)

3.1

struct **bHYPRE_Vector__object**

Symbol "bHYPRE.Vector" (version 1.0.0)

3.2

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.

4

Matrices and Vectors

Names

4.1

IJParCSR Matrix

Names

void
Constructor function for the class

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__createRemote (const char *,
sidl_BaseInterface *_ex)
RMI constructor function for the class

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__connect (const char *, sidl_BaseInterface *_ex)
RMI connector function for the class

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix_Create (bHYPRE_MPICommunicator
mpi_comm, int32_t ilower,
int32_t iupper, int32_t jlower,
int32_t jupper)
Method: Create[]

bHYPRE_IJParCSRMatrix

bHYPRE_IJParCSRMatrix_GenerateLaplacian (bHYPRE_MPICommunicator mpi_comm, int32_t nx, int32_t ny, int32_t nz, int32_t Px, int32_t Py, int32_t Pz, int32_t p, int32_t q, int32_t r, double* values, int32_t nvalues, int32_t discretization)

Method: GenerateLaplacian[]

4.1.2 int32_t
bHYPRE_IJParCSRMatrix_SetDiagOffdSizes (bHYPRE_IJParCSRMatrix self, int32_t* diag_sizes, int32_t* offdiag_sizes, int32_t local_nrows)

(Optional) Set the max number of nonzeros to expect in each row of the diagonal and off-diagonal blocks 22

4.1.3 int32_t
bHYPRE_IJParCSRMatrix_SetCommunicator (bHYPRE_IJParCSRMatrix self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator 22

int32_t
bHYPRE_IJParCSRMatrix_Initialize (bHYPRE_IJParCSRMatrix self)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.1.4 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 23

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 23

4.1.6 int32_t
bHYPRE_IJParCSRMatrix_SetValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

Sets values for `nrows` *of the matrix* 24

4.1.7 int32_t

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

Set the int parameter associated with `name`

int32_t
bHYPRE_IJParCSRMatrix_SetDoubleParameter (bHYPRE_IJParCSRMatrix self, const char* name, double value)

Set the double parameter associated with `name`

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

Set the string parameter associated with `name`

int32_t
bHYPRE_IJParCSRMatrix_SetIntArray1Parameter (bHYPRE_IJParCSRMatrix self, const char* name, int32_t* value, int32_t nvalues)

Set the int 1-D array parameter associated with `name`

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, double* value, int32_t nvalues)

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`

int32_t
bHYPRE_IJParCSRMatrix_ApplyAdjoint (bHYPRE_IJParCSRMatrix self, bHYPRE_Vector b,
bHYPRE_Vector* x)

Apply the adjoint of the operator to `b`, *returning* `x`

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

void
bHYPRE_IJParCSRMatrix__exec (bHYPRE_IJParCSRMatrix self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)

Select and execute a method by name

void
bHYPRE_IJParCSRMatrix__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)

static Exec method for reflexity

char*
bHYPRE_IJParCSRMatrix__getURL (bHYPRE_IJParCSRMatrix self)

Get the URL of the Implementation of this object (for RMI)

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 IJMatrixView, Operator, or CoefficientAccess objects using the `__cast` methods.

4.1.2

int32_t
bHYPRE_IJParCSRMatrix_SetDiagOffdSizes (bHYPRE_IJParCSRMatrix self, int32_t* diag_sizes, int32_t* offdiag_sizes, int32_t local_nrows)

(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_SetCommunicator (bHYPRE_IJParCSRMatrix self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, Use Create()

4.1.4

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.

4.1.5

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, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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`. The last argument is the size of the cols and values arrays, i.e. the total number of nonzeros being provided, i.e. the sum of all values in ncols. This functin 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, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros)

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, int32_t* sizes, int32_t nrows)

(Optional) Set the max number of nonzeros to expect in each row. The array `sizes` contains estimated sizes for each row on this process. The integer nrows is the number of rows in the local matrix. 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, bHYPRE_MPICommunicator 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**
Symbol "bHYPRE 29

void
Constructor function for the class

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector__createRemote (const char *, sidl_BaseInterface *_ex)
RMI constructor function for the class

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector__connect (const char *, sidl_BaseInterface *_ex)
RMI connector function for the class

bHYPRE_IJParCSRVector

bHYPRE_IJParCSRVector_Create (bHYPRE_MPICommunicator mpi_comm, int32_t jlower, int32_t jupper)

Method: Create[]

4.2.2 int32_t
bHYPRE_IJParCSRVector_SetCommunicator (bHYPRE_IJParCSRVector self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator 29

int32_t
bHYPRE_IJParCSRVector_Initialize (bHYPRE_IJParCSRVector self)

Prepare an object for setting coefficient values, whether for the first time or subsequently

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 29

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

Set the local range for a vector object 30

4.2.5 int32_t
bHYPRE_IJParCSRVector_SetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values)

Sets values in vector 30

4.2.6 int32_t
bHYPRE_IJParCSRVector_AddToValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values)

Adds to values in vector 30

int32_t
bHYPRE_IJParCSRVector_GetLocalRange (bHYPRE_IJParCSRVector self, int32_t* jlower, int32_t* jupper)

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

4.2.7 int32_t
bHYPRE_IJParCSRVector_GetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values)

Gets values in vector 31

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

Print the vector to file 31

4.2.9 int32_t

int32_t
bHYPRE_IJParCSRVector_Read (bHYPRE_IJParCSRVector self,
const char* filename,
bHYPRE_MPICommunicator comm)
Read the vector from file 31

int32_t
bHYPRE_IJParCSRVector_Clear (bHYPRE_IJParCSRVector self)
Set `self` *to 0*

int32_t
bHYPRE_IJParCSRVector_Copy (bHYPRE_IJParCSRVector self,
bHYPRE_Vector x)
Copy x into `self`

4.2.10 int32_t
bHYPRE_IJParCSRVector_Clone (bHYPRE_IJParCSRVector self,
bHYPRE_Vector* x)
Create an `x` *compatible with* `self` 32

int32_t
bHYPRE_IJParCSRVector_Scale (bHYPRE_IJParCSRVector self,
double a)
Scale `self` *by* `a`

int32_t
bHYPRE_IJParCSRVector_Dot (bHYPRE_IJParCSRVector self,
bHYPRE_Vector x, double* d)
Compute `d`, *the inner-product of* `self` *and* `x`

int32_t
bHYPRE_IJParCSRVector_Axpy (bHYPRE_IJParCSRVector self,
double a, bHYPRE_Vector x)
Add `a*x` *to* `self`

obj
Cast method for interface and class type conversions

void*
bHYPRE_IJParCSRVector__cast2 (void* obj, const char* type)
String cast method for interface and class type conversions

void
bHYPRE_IJParCSRVector__exec (bHYPRE_IJParCSRVector self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_IJParCSRVector__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_IJParCSRVector__getURL (bHYPRE_IJParCSRVector self)
Get the URL of the Implementation of this object (for RMI)

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 IJVectorView or Vector objects using the `__cast` methods.

4.2.2

int32_t
bHYPRE_IJParCSRVector_SetCommunicator (bHYPRE_IJParCSRVector self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, Use Create()

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_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.

4.2.5

int32_t
bHYPRE_IJParCSRVector_SetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* 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.

4.2.6

int32_t
bHYPRE_IJParCSRVector_AddToValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values)

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

Not collective.

4.2.7

int32_t
bHYPRE_IJParCSRVector_GetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values)

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

Not collective.

4.2.8

int32_t
bHYPRE_IJParCSRVector_Print (bHYPRE_IJParCSRVector self, const char* filename)

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

4.2.9

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

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

4.2.10

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.

5

Solver Interface

Names

5.1 struct **bHYPRE_Solver__object**
Symbol "bHYPRE 34

extern C bHYPRE_Solver
bHYPRE_Solver__connect (const char *, sidl_BaseInterface *_ex)
RMI connector function for the class

5.2 int32_t
bHYPRE_Solver_SetOperator (bHYPRE_Solver self,
bHYPRE_Operator A)
Set the operator for the linear system being solved 34

5.3 int32_t
bHYPRE_Solver_SetTolerance (bHYPRE_Solver self, double tolerance)
(Optional) Set the convergence tolerance 34

5.4 int32_t
bHYPRE_Solver_SetMaxIterations (bHYPRE_Solver self,
int32_t max_iterations)
(Optional) Set maximum number of iterations 35

5.5 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 35

5.6 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 35

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

struct bHYPRE_Solver__object* 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

void

bHYPRE_Solver__exec (bHYPRE_Solver self, const char* methodName,
sidl_io_Deserializer inArgs, sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_Solver__sexec (const char* methodName,
sidl_io_Deserializer inArgs, sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_Solver__getURL (bHYPRE_Solver self)
Get the URL of the Implementation of this object (for RMI)

5.1

struct **bHYPRE_Solver__object**

Symbol "bHYPRE.Solver" (version 1.0.0)

5.2

int32_t
bHYPRE_Solver_SetOperator (bHYPRE_Solver self, bHYPRE_Operator A)

Set the operator for the linear system being solved. DEPRECATED. use Create

5.3

int32_t
bHYPRE_Solver_SetTolerance (bHYPRE_Solver self, double tolerance)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

5.4

int32_t
bHYPRE_Solver_SetMaxIterations (bHYPRE_Solver self, int32_t max_iterations)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

5.5

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`. DEPRECATED use SetIntParameter

5.6

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`. DEPRECATED use SetIntParameter

6 ParCSR Solvers

Names

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

6.1 ParCSRDiagScale Solver

Names

bHYPRE_ParCSRDiagScale_SetCommunicator (
bHYPRE_ParCSRDiagScale self,
bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator 40

int32_t
bHYPRE_ParCSRDiagScale_SetIntParameter (
bHYPRE_ParCSRDiagScale self, const char* name,
int32_t value)

Set the int parameter associated with `name`

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,
int32_t* value,
int32_t nvalues)

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,
double* value,
int32_t nvalues)

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)

Set the int parameter associated with `name`

int32_t
bHYPRE_ParCSRDiagScale_GetDoubleValue (bHYPRE_ParCSRDiagScale self, const char* name, double* value)

Get the double parameter associated with `name`

int32_t
bHYPRE_ParCSRDiagScale_Setup (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector x)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

int32_t
bHYPRE_ParCSRDiagScale_Apply (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector* x)

Apply the operator to `b`, *returning* `x`

int32_t
bHYPRE_ParCSRDiagScale_ApplyAdjoint (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector* x)

Apply the adjoint of the operator to `b`, *returning* `x`

6.1.3 int32_t
bHYPRE_ParCSRDiagScale_SetOperator (bHYPRE_ParCSRDiagScale self, bHYPRE_Operator A)

Set the operator for the linear system being solved 40

6.1.4 int32_t
bHYPRE_ParCSRDiagScale_SetTolerance (bHYPRE_ParCSRDiagScale self, double tolerance)

(Optional) Set the convergence tolerance 40

6.1.5 int32_t
bHYPRE_ParCSRDiagScale_SetMaxIterations (bHYPRE_ParCSRDiagScale self, int32_t max_iterations)

(Optional) Set maximum number of iterations 41

6.1.6 int32_t

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_SetCommunicator (
bHYPRE_ParCSRDiagScale self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, use Create:

6.1.3

int32_t
bHYPRE_ParCSRDiagScale_SetOperator (bHYPRE_ParCSRDiagScale self,
bHYPRE_Operator A)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.1.4

int32_t
bHYPRE_ParCSRDiagScale_SetTolerance (bHYPRE_ParCSRDiagScale self,
double tolerance)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.1.5

int32_t
bHYPRE_ParCSRDiagScale_SetMaxIterations (
bHYPRE_ParCSRDiagScale self, int32_t max_iterations)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.1.6

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`. DEPRECATED use SetIntParameter

6.1.7

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`. DEPRECATED use SetIntParameter

6.2

ParCSR BoomerAMG Solver

Names

6.2.1 struct **bHYPRE_BoomerAMG__object**

Symbol "bHYPRE 45

void

Constructor function for the class

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG__createRemote (const char *,
sidl_BaseInterface *_ex)

RMI constructor function for the class

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG__connect (const char *, sidl_BaseInterface *_ex)

RMI connector function for the class

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_IJParCSRMatrix A)

Method: Create[]

int32_t
bHYPRE_BoomerAMG_SetLevelRelaxWt (bHYPRE_BoomerAMG self,
double relax_wt, int32_t level)

Method: SetLevelRelaxWt[]

int32_t
bHYPRE_BoomerAMG_InitGridRelaxation (bHYPRE_BoomerAMG self,
struct sidl_int__array**
num_grid_sweeps,
struct sidl_int__array**
grid_relax_type,
struct sidl_int__array**
grid_relax_points,
int32_t coarsen_type,
struct sidl_double__array**
relax_weights,
int32_t max_levels)

Method: InitGridRelaxation[]

6.2.2 int32_t
bHYPRE_BoomerAMG_SetCommunicator (bHYPRE_BoomerAMG self,
bHYPRE_MPICommunicator
mpi_comm)

Set the MPI Communicator 47

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,
int32_t* value,
int32_t nvalues)

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,
double* value,
int32_t nvalues)

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)

Apply the operator to `b`, *returning* `x`

int32_t
bHYPRE_BoomerAMG_ApplyAdjoint (bHYPRE_BoomerAMG self,
bHYPRE_Vector b,
bHYPRE_Vector* x)

Apply the adjoint of the operator to `b`, *returning* `x`

6.2.3 int32_t
bHYPRE_BoomerAMG_SetOperator (bHYPRE_BoomerAMG self,
bHYPRE_Operator A)

Set the operator for the linear system being solved 47

6.2.4 int32_t
bHYPRE_BoomerAMG_SetTolerance (bHYPRE_BoomerAMG self,
double tolerance)

(Optional) Set the convergence tolerance 47

6.2.5 int32_t
bHYPRE_BoomerAMG_SetMaxIterations (bHYPRE_BoomerAMG self,
int32_t max_iterations)

(Optional) Set maximum number of iterations 48

6.2.6 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 48

6.2.7 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 48

int32_t
bHYPRE_BoomerAMG_GetNumIterations (bHYPRE_BoomerAMG self,
int32_t* num_iterations)

(Optional) Return the number of iterations taken

int32_t
bHYPRE_BoomerAMG_GetRelResidualNorm (bHYPRE_BoomerAMG
self, double* norm)

(Optional) Return the norm of the relative residual

struct bHYPRE_BoomerAMG__object* bHYPRE_BoomerAMG__cast void* **obj**

Cast method for interface and class type conversions

void*
bHYPRE_BoomerAMG__cast2 (void* obj, const char* type)
String cast method for interface and class type conversions

void
bHYPRE_BoomerAMG__exec (bHYPRE_BoomerAMG self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_BoomerAMG__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_BoomerAMG__getURL (bHYPRE_BoomerAMG self)
Get the URL of the Implementation of this object (for RMI)

6.2.1 struct **bHYPRE_BoomerAMG__object**

Symbol "bHYPRE.BoomerAMG" (version 1.0.0)

Algebraic multigrid solver, based on classical Ruge-Stueben.

BoomerAMG requires an IJParCSR matrix

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. DEPRECATED: Use NumSweeps or Cycle?NumSweeps instead.

NumSweeps (`Int`) - number of sweeps for fine grid, up and down cycle.

Cycle0NumSweeps (`Int`) - number of sweeps for fine grid

Cycle1NumSweeps (`Int`) - number of sweeps for down cycle

Cycle2NumSweeps (`Int`) - number of sweeps for up cycle

Cycle3NumSweeps (`Int`) - number of sweeps for coarse grid

GridRelaxType (`IntArray 1D`) - type of smoother used on fine and coarse grid, up and down cycle. DEPRECATED: Use RelaxType or Cycle?RelaxType instead.

RelaxType (`Int`) - type of smoother for fine grid, up and down cycle.

Cycle0RelaxType (`Int`) - type of smoother for fine grid

Cycle1RelaxType (`Int`) - type of smoother for down cycle

Cycle2RelaxType (`Int`) - type of smoother for up cycle

Cycle3RelaxType (`Int`) - type of smoother for coarse grid

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

RelaxWeight (`DoubleArray 1D`) - relaxation weight for smoothed Jacobi and hybrid SOR. DEPRECATED: Instead, use the RelaxWt parameter and the SetLevelRelaxWt function.

RelaxWt (`Int`) - relaxation weight for all levels 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`) -

The following function is specific to this class:

SetLevelRelxWeight (`Double , Int`) - relaxation weight for one specified level of smoothed Jacobi and hybrid SOR.

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

6.2.2

int32_t
bHYPRE_BoomerAMG_SetCommunicator (bHYPRE_BoomerAMG self, bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, use Create:

6.2.3

int32_t
bHYPRE_BoomerAMG_SetOperator (bHYPRE_BoomerAMG self, bHYPRE_Operator A)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.2.4

int32_t
bHYPRE_BoomerAMG_SetTolerance (bHYPRE_BoomerAMG self, double tolerance)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.2.5

int32_t
bHYPRE_BoomerAMG_SetMaxIterations (bHYPRE_BoomerAMG self, int32_t max_iterations)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.2.6

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`. DEPRECATED use SetIntParameter

6.2.7

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`. DEPRECATED use SetIntParameter

7 PreconditionedSolver Interface

Names

extern C bHYPRE_PreconditionedSolver
bHYPRE_PreconditionedSolver__connect (const char *,
sidl_BaseInterface *_ex)
RMI connector function for the class

int32_t
bHYPRE_PreconditionedSolver_SetPreconditioner (
bHYPRE_PreconditionedSolver
self,
bHYPRE_Solver s)
Set the preconditioner

int32_t
bHYPRE_PreconditionedSolver_GetPreconditioner (
bHYPRE_PreconditionedSolver
self,
bHYPRE_Solver* s)
Method: GetPreconditioner[]

int32_t
bHYPRE_PreconditionedSolver_Clone (bHYPRE_PreconditionedSolver
self,
bHYPRE_PreconditionedSolver* x)
Method: Clone[]

obj
Cast method for interface and class type conversions

void*
bHYPRE_PreconditionedSolver__cast2 (void* obj, const char* type)
String cast method for interface and class type conversions

void
bHYPRE_PreconditionedSolver__exec (bHYPRE_PreconditionedSolver self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_PreconditionedSolver__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
static Exec method for reflexity

char*

bHYPRE_PreconditionedSolver__getURL (bHYPRE_PreconditionedSolver self)
Get the URL of the Implementation of this object (for RMI)

7.1

struct **bHYPRE_PreconditionedSolver__object**

Symbol "bHYPRE.PreconditionedSolver" (version 1.0.0)

8 Preconditioned Solvers

Names

8.1 PCG Preconditioned Solver

Names

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, int32_t* value,
int32_t nvalues)

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)

Set the int 2-D array parameter associated with `name`

int32_t
bHYPRE_PCG_SetDoubleArray1Parameter (bHYPRE_PCG self,
const char* name,
double* value,
int32_t nvalues)

Set the double 1-D array parameter associated with `name`

int32_t
bHYPRE_PCG_SetDoubleArray2Parameter (bHYPRE_PCG self,
const char* name, struct
sidl_double__array* value)

Set the double 2-D array parameter associated with `name`

int32_t
bHYPRE_PCG_GetIntValue (bHYPRE_PCG self, const char* name,
int32_t* value)

Set the int parameter associated with `name`

int32_t
bHYPRE_PCG_GetDoubleValue (bHYPRE_PCG self, const char* name,
double* value)

Get the double parameter associated with `name`

int32_t
bHYPRE_PCG_Setup (bHYPRE_PCG self, bHYPRE_Vector b,
bHYPRE_Vector x)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

int32_t
bHYPRE_PCG_Apply (bHYPRE_PCG self, bHYPRE_Vector b,
bHYPRE_Vector* x)

Apply the operator to `b`, *returning* `x`

int32_t

bHYPRE_PCG_ApplyAdjoint (bHYPRE_PCG self, bHYPRE_Vector b, bHYPRE_Vector* x)

Apply the adjoint of the operator to `b`*, returning* `x`

8.1.3 int32_t

bHYPRE_PCG_SetOperator (bHYPRE_PCG self, bHYPRE_Operator A)

Set the operator for the linear system being solved 54

8.1.4 int32_t

bHYPRE_PCG_SetTolerance (bHYPRE_PCG self, double tolerance)

(Optional) Set the convergence tolerance 55

8.1.5 int32_t

bHYPRE_PCG_SetMaxIterations (bHYPRE_PCG self, int32_t max_iterations)

(Optional) Set maximum number of iterations 55

8.1.6 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 55

8.1.7 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 56

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

int32_t

bHYPRE_PCG_GetPreconditioner (bHYPRE_PCG self, bHYPRE_Solver* s)

Method: GetPreconditioner[]

int32_t

bHYPRE_PCG_Clone (bHYPRE_PCG self, bHYPRE_PreconditionedSolver* x)

Method: Clone[]

struct bHYPRE_PCG__object* 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

void

bHYPRE_PCG__exec (bHYPRE_PCG self, const char* methodName,
sidl_io_Deserializer inArgs, sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_PCG__sexec (const char* methodName,
sidl_io_Deserializer inArgs, sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_PCG__getURL (bHYPRE_PCG self)
Get the URL of the Implementation of this object (for RMI)

8.1.1

struct **bHYPRE_PCG__object**

Symbol "bHYPRE.PCG" (version 1.0.0)

8.1.2

int32_t
bHYPRE_PCG_SetCommunicator (bHYPRE_PCG self,
bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, use Create:

8.1.3

int32_t
bHYPRE_PCG_SetOperator (bHYPRE_PCG self, bHYPRE_Operator A)

Set the operator for the linear system being solved. DEPRECATED. use Create

8.1.4

int32_t **bHYPRE_PCG_SetTolerance** (bHYPRE_PCG self, double tolerance)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

8.1.5

int32_t
bHYPRE_PCG_SetMaxIterations (bHYPRE_PCG self, int32_t max_iterations)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

8.1.6

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`. DEPRECATED use SetIntParameter

8.1.7

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`. DEPRECATED use SetIntParameter

8.2

GMRES Preconditioned Solver

Names

bHYPRE_GMRES_SetDoubleParameter (bHYPRE_GMRES self,
const char* name, double value)

Set the double parameter associated with `name`

int32_t
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,
int32_t* value,
int32_t nvalues)

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,
double* value,
int32_t nvalues)

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_ApplyAdjoint (bHYPRE_GMRES self,
bHYPRE_Vector b, bHYPRE_Vector* x)
Apply the adjoint of the operator to `b`, *returning* `x`

8.2.3 int32_t
bHYPRE_GMRES_SetOperator (bHYPRE_GMRES self,
bHYPRE_Operator A)
Set the operator for the linear system being solved 60

8.2.4 int32_t
bHYPRE_GMRES_SetTolerance (bHYPRE_GMRES self,
double tolerance)
(Optional) Set the convergence tolerance 60

8.2.5 int32_t
bHYPRE_GMRES_SetMaxIterations (bHYPRE_GMRES self,
int32_t max_iterations)
(Optional) Set maximum number of iterations 60

8.2.6 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 61

8.2.7 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 61

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

int32_t
bHYPRE_GMRES_GetPreconditioner (bHYPRE_GMRES self,
bHYPRE_Solver* s)
Method: GetPreconditioner[]

int32_t
bHYPRE_GMRES_Clone (bHYPRE_GMRES self,
bHYPRE_PreconditionedSolver* x)
Method: Clone[]

struct bHYPRE_GMRES__object* 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

void
bHYPRE_GMRES__exec (bHYPRE_GMRES self,
const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
Select and execute a method by name

void
bHYPRE_GMRES__sexec (const char* methodName,
sidl_io_Deserializer inArgs,
sidl_io_Serializer outArgs)
static Exec method for reflexity

char*
bHYPRE_GMRES__getURL (bHYPRE_GMRES self)
Get the URL of the Implementation of this object (for RMI)

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.

The regular GMRES solver calls Babel-interface matrix and vector functions. The HGMRES solver calls HYPRE interface functions. The regular solver will work with any consistent matrix, vector, and preconditioner classes. The HGMRES solver will work with the more common combinations.

The HGMRES solver checks whether the matrix, vectors, and preconditioner are of known types, and will not work with any other types. Presently, the recognized data types are: matrix, vector: IJParCSRMatrix, IJParCSRVector preconditioner: BoomerAMG, ParCSRDiagScale

8.2.2

int32_t
bHYPRE_GMRES_SetCommunicator (bHYPRE_GMRES self,
bHYPRE_MPICommunicator mpi_comm)

Set the MPI Communicator. DEPRECATED, use Create:

8.2.3

int32_t
bHYPRE_GMRES_SetOperator (bHYPRE_GMRES self, bHYPRE_Operator A)

Set the operator for the linear system being solved. DEPRECATED. use Create

8.2.4

int32_t
bHYPRE_GMRES_SetTolerance (bHYPRE_GMRES self, double tolerance)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

8.2.5

int32_t
bHYPRE_GMRES_SetMaxIterations (bHYPRE_GMRES self, int32_t max_iterations)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

8.2.6

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`. DEPRECATED use SetIntParameter

8.2.7

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`. DEPRECATED use SetIntParameter

Class Graph