TECHNISCHE UNIVERSITÄT
TUHH
HAMBURG-HARBURG

Berichte des
Forschungsschwerpunktes
Informations- und
Kommunikationstechnik

Olaf Knüppel

BIAS — Basic Interval Arithmetic Subroutines

Juli 1993

Bericht 93.3

Olaf Knüppel: Juli 1993

BIAS — Basic Interval Arithmetic Subroutines

Bericht 93.3

IMPORTANT NOTE:

Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

Prof. Dr. S. M. Rump
Technische Universität Hamburg-Harburg
Technische Informatik III
D-21071 Hamburg
Germany

This report is available
via anonymous ftp from:
`ti3sun.ti3.tu-harburg.de`
in the file:
`/pub/reports/report93.3.ps.Z`

BIAS — Basic Interval Arithmetic Subroutines

Olaf Knüppel*
Technische Informatik III
Technische Universität Hamburg-Harburg
email: knueppel@tu-harburg.d400.de

July 15, 1993

Abstract

In numerical linear algebra the set of commonly used FORTRAN callable functions for Basic Linear Algebra Subroutines (BLAS) is very popular. The idea of BLAS is to provide an interface for basic vector and matrix operations with specific and fast implementations on various machines, the latter frequently provided by the manufacturers. Along these lines we present a specification for Basic Interval Arithmetic Subroutines (BIAS). The objective is

- portablity,
- no dependency on a specific interval representation,
- very efficient use of the underlying hardware.

The routines cover the basic scalar, vector, and matrix operations. The set of routines defined by this specification can easily be included in higher-level, user-friendly programming environments, e.g. written in C++.

*D-21071 Hamburg

Contents

1 Introduction **3**

2 Data Types **4**

2.1 Notation 5

2.2 Implementation Dependencies 5

3 Initialization **6**

4 Variables **6**

5 Scalar Operations — `Bias0` **6**

6 Vector Operations — `Bias1` **12**

7 Matrix Operations — `Bias2` **20**

8 Scalar Standard Functions — `BiasF` **28**

1 Introduction

This is a short description of the idea and the current implementation of BIAS (Basic Interval Arithmetic Subroutines). It is assumed that the reader is familiar with the basics of interval mathematics.

The developement of BIAS was guided by the ideas of BLAS, i.e. to provide an interface for basic vector and matrix operations with specific and fast implementations on various machines, the latter frequently provided by the manufactures. The idea of BIAS is to give such an interface for interval operations with the objective:

- portability
- independency of a specific interval representation
- very efficient use of the unterlying hardware

First, we describe the data types and variables supported by BIAS, then after giving some notations we describe all routines from BIAS. The routines are described in the groups: scalar operations, vector operations, and matrix operations. At last, scalar interval standard functions are described. We give the header and a short description of each routine.

Currenly, three implementations of BIAS exist, all of them use the same interval representation as it will be described below. The three implementations are in particular:

1. Using the directed rounding. Because the switching of the rounding mode is machine dependent and not available on all architectures, this implementation is the most hardware dependent one.

   Currently, it is available for IBM RS/6000 architectures, Sparc architectures, 680x0 machines, and PCs (80386 and higher) with coprocessor. In all cases, assembler routines for switching the rounding mode are provided and it has been taken care to avoid unneccessary switching of the rounding mode.

2. Using an estimation of the error which occurs in performing a floating-point operation. The error estimation is used to change the floating-point calculated bounds in such a way that the isotonicity of the interval operations is always fulfilled.

   This implementation uses only standard floating-point operations and because of this it can be used for a variety of machines.

   The only machine dependency lies in the maximal possible relative error of the floating-point operations. In the current implementation it is assumed, that the following for

the true result $\widehat{x}$ and the computed result $\widetilde{x}$ holds:

$$(1-\epsilon)\widetilde{x} - \eta \leq \widehat{x} \leq (1+\epsilon)\widetilde{x} + \eta \text{ for } \widetilde{x} \geq 0$$
$$(1+\epsilon)\widetilde{x} - \eta \leq \widehat{x} \leq (1-\epsilon)\widetilde{x} + \eta \text{ for } \widetilde{x} < 0$$

Where ϵ is the machine precision and η the smallest positive machine number (see variables below).

3. Using the computed floating-point result as bounds without any modification. This implementation is very fast, but doesn't always give correct bounds. The main application is in cases where a least significant bit accuracy is not needed, e.g. in global optimization, where only large intervals are used.

2 Data Types

The following data types are defined and used by BIAS:

Data Type	Description
`REAL`	The real type (either `double` or `float`).
`PREAL`	Pointer to `REAL`
`INT`	The integer type
`BIASINTERVAL`	The interval type
`PBIASINTERVAL`	Pointer to `BIASINTERVAL`

In all current implementations of BIAS, the following definitions are given for the data types above:

```
typedef INT    int;
typedef REAL   double;
typedef PREAL  * double;
typedef struct {
  REAL inf;
  REAL sup;
} BIASINTERVAL;
typedef BIASINTERVAL * BIASINTERVAL;
```

Vectors are stored blockwise, beginning with the first component. A vector is described by two values: A pointer to the elements (`PREAL` for real and `PBIASINTERVAL` for interval vectors) and the number of elements (`INT`).

Matrices are stored rowwise in a block (that is the normal C storage, or the transposed FORTRAN storage). A matrix is described by three values: A pointer to the elements (`PREAL` for real and `PBIASINTERVAL` for interval matrices), the number of rows (`INT`), and the number of columns (`INT`).

2.1 Notation

In program text (e.g. in routine headers), we denote interval parameters by capital and real parameters by small letters. If preceeded by a small `p`, the parameters are pointers, which point to the objects without the leading `p`. If not stated explicitly otherwise, all parameters are scalar types.

The naming conventions for the routines in BIAS are as follows: All routine names start with `Bias`. After that, the purpose of the routine follows (e.g. `Mul` for multiplication). Last, the main parameter description is appended, where the letter `R` denotes a real and the letter `I` an interval parameter. Vectors and matrix parameters are given by using the letter `V` or `M` as a prefix to the data type. For example: The addition of a real and an interval vector is coded as `BiasAddVRVI`.

2.2 Implementation Dependencies

Depending on the implementation, some limitations may occur due to speed reasons. For example in the current implementation the operands and the result of an interval subtraction must not be the same.

In general, the following assumption holds:

> The result of any BIAS operation, i.e. any value that is changed by a routine from BIAS should *not* be the same as any parameter of that routine.

Another implementation dependency is the treatement of exceptions (e.g. division by zero). In the current implementation, one method out of two can be chosen at compile time:

1. A division by zero is always reported and the program is terminated.
2. If possible, the result is silently set to e.g. to $[-\infty, +\infty]$. Otherwise the error is reported and the program is terminated.

Additionally, when using directed roundings, the rounding mode direction is undefined after execution of any routine from BIAS. In general, the rounding mode is set to upward rounding.

If neccessary (e.g. due to sensitive floating point expressions), BIAS can be configured to set the rounding mode to nearest after execution.

The controlling parameters for the configuration of the behaviour of BIAS are contained and explained in the file `BiasInt.h`.

In the routine description, expressions like `R = A + B` where `A`,`B`,`R` are intervals should be understand as

$$\mathtt{R} \supseteq \{\, a + b \mid a \in \mathtt{A},\ b \in \mathtt{B} \,\}$$

except for implementations where the computed bounds are not guaranteed to be correct. In this case, the above expression should be read as:

$$\mathtt{R} \approx \{\, a + b \mid a \in \mathtt{A},\ b \in \mathtt{B} \,\}$$

3 Initialization

All neccessary initializations for BIAS are performed by a call to `BiasInit()`.

4 Variables

In this section we describe the variables with machine dependent constants which are supported by BIAS. Depending on the implementation, some of these variables may not be available. At least, `BiasEpsilon` and `BiasEta` should always contain reliable values.

Variable	Value
`BiasEpsilon`	The machine precision $\min_{x \in \mathbb{F}} \{\, x \geq 0 \mid 1 + x > 1 \,\}$
`BiasEta`	The smallest positive floating-point number $\min_{x \in \mathbb{F}} \{x > 0\}$
`BiasNaN`	NaN (not a number)
`BiasPosInf`	$+\infty$
`BiasNegInf`	$-\infty$

We denote by $\mathbb{F}$ the set of floating-point numbers.

5 Scalar Operations — `Bias0`

Headerfile: `Bias0.h`

`BiasInit()`

Initializes all internal values, must be called prior any other BIAS call.

`BiasRoundUp()`

Switches the rounding mode to upwards. Generates an error message, if no rounding mode switching is supported.

`BiasRoundDown()`

Switches the rounding mode to downwards. Generates an error message, if no rounding mode switching is supported.

`BiasRoundNear()`

Switches the rounding mode to nearest. Generates an error message, if no rounding mode switching is supported.

`REAL BiasPredR(PREAL pa)`

Returns the predecessor of `a`.

`REAL BiasSuccR(PREAL pa)`

Returns the successor of `a`.

`BiasPredI(PBIASINTERVAL pR, PBIASINTERVAL pA)`

On return, `R` is the largest interval which is contained in the interior of `A`.

`BiasSuccI(PBIASINTERVAL pR, PBIASINTERVAL pA)`

On return, `R` is the smallest interval, where `A` is contained in the interior.

`BiasAddRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Calculates the interval `R` which contains the true result of `a` + `b`.

`BiasAddRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Calculates R = a + B.

`BiasAddIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Calculates R = A + b.

`BiasAddII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates R = A + B.

`BiasNeg(PBIASINTERVAL pR, PINTERVAL pA)`

Calculates R = −A.

`BiasSubRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Calculates the interval R which contains the true result of a − b.

`BiasSubRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Calculates R = a − B.

`BiasSubIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Calculates R = A − b.

`BiasSubII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates R = A − B.

`BiasMulRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Calculates the interval R which contains the true result of a · b.

`BiasMulRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \mathtt{a} \cdot \mathtt{B}$.

`BiasMulIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Calculates $\mathtt{R} = \mathtt{A} \cdot \mathtt{b}$.

`BiasMulII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \mathtt{A} \cdot \mathtt{B}$.

`BiasDivRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Calculates the interval `R` which contains the true result of $\frac{\mathtt{a}}{\mathtt{b}}$.

`BiasDivRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \frac{\mathtt{a}}{\mathtt{B}}$.

`BiasDivIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Calculates $\mathtt{R} = \frac{\mathtt{A}}{\mathtt{b}}$.

`BiasDivII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \frac{\mathtt{A}}{\mathtt{B}}$.

`BiasMacRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Adds to the interval `R` an interval which contains the true result of $\mathtt{a} \cdot \mathtt{b}$.

`BiasMacRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \mathtt{R} + \mathtt{a} \cdot \mathtt{B}$.

`BiasMacIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Calculates $\mathtt{R} = \mathtt{R} + \mathtt{A} \cdot \mathtt{b}$.

`BiasMacII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates $\mathtt{R} = \mathtt{R} + \mathtt{A} \cdot \mathtt{B}$.

`REAL BiasInf(PBIASINTERVAL pA)`

Returns the lower bound of the interval `A`.

`REAL BiasSup(PBIASINTERVAL pA)`

Returns the upper bound of the interval `A`.

`REAL BiasMid(PBIASINTERVAL pA)`

Returns the midpoint of the interval `A`.

`BiasMidRad(PREAL pm, PREAL pr, PBIASINTERVAL pA)`

Caclulates the midpoint `m` and the radius `r` of the interval `A`.

`REAL BiasDiam(PBIASINTERVAL pA)`

Returns the diameter of the interval `A`.

`REAL BiasAbs(PBIASINTERVAL pA)`

Returns the absolute value of the interval `A`, i.e. $\max\{|\underline{\mathtt{A}}|, |\overline{\mathtt{A}}|\}$.

`REAL BiasDistRI(PREAL a, PBIASINTERVAL pB)`

Returns the distance between `a` and `B`. If $\mathtt{a} \in \mathtt{B}$ then 0 is returned.

`REAL BiasDistII(PBIASINTERVAL pA, PBIASINTERVAL pB)`

Returns the distance between `A` and `B`.

`INT BiasIntersection(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Calculates the intersection `R` between `A` and `B`. If the intersection is empty, a value of 0 is returned and the contents of `R` are undefined. Otherwise, a value of 1 is returned.

`BiasHullR(PBIASINTERVAL pR, PREAL pa)`

$\mathtt{R} = [\mathtt{a}, \mathtt{a}]$.

`BiasHullRR(PBIASINTERVAL pR, PREAL pa, PREAL pb)`

Initializes `R` with the convex hull of `a` and `b`.

`BiasHullRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB)`

Initializes `R` with the convex hull of `a` and `B`.

`BiasHullIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb)`

Initializes `R` with the convex hull of `A` and `b`.

`BiasHullII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Initializes `R` with the convex hull of `A` and `B`.

`INT BiasInR(PREAL pa, PBIASINTERVAL pB)`

Returns 1, if $\mathtt{a} \in \mathtt{B}$. Otherwise, 0 is returned.

`INT BiasInI(PBIASINTERVAL pA, PBIASINTERVAL pB)`

Returns 1, if $\mathtt{A} \subseteq \mathtt{B}$. Otherwise, 0 is returned.

`INT BiasInInteriorR(PREAL pa, PBIASINTERVAL pB)`

Returns 1, if a $\in$ Interior(B). Otherwise, 0 is returned.

`INT BiasInInteriorI(PBIASINTERVAL pA, PBIASINTERVAL pB)`

Returns 1, if A $\subseteq$ Interior(B). Otherwise, 0 is returned.

6 Vector Operations — Bias1

Headerfile: Bias1.h

In all vector routines the parameter dim denotes the number of vector elements. Throughout this description, elements are numbered beginning with the index 1. Vector parameters are given as pointer to the elements where the pointer points to the first element. We denote a single vector element by using indexing.

`VOID BiasPredVR(PREAL pr, PREAL pa, INT dim)`

The vector r is set to the predecessor of the vector a.

$$\mathtt{r}_i = \mathrm{pred}(\mathtt{a}_i), \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasSuccVR(PREAL pr, PREAL pa, INT dim)`

The vector r is set to the successor of the vector a.

$$\mathtt{r}_i = \mathrm{pred}(\mathtt{a}_i), \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasPredVI(PBIASINTERVAL pR, PBIASINTERVAL pA, INT dim)`

Each component of the resulting interval vector R is set to the largest interval which is contained in the interior of the appropriate component of the interval vector A.

`VOID BiasSuccVI(PBIASINTERVAL pR, PBIASINTERVAL pA, INT dim)`

Each component of the resulting interval vector R is set to the smallest interval where the appropriate component of the interval vector A is contained in the interior.

`VOID BiasAddVRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)`

Calculates the interval vector `R` which contains the true result of the sum of the vectors `a` and `b`.

$$\mathtt{R}_i = \mathtt{a}_i + \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasAddVRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)`

Calculates the interval vector `R` as the sum of the vector `a` and the interval vector `B`.

$$\mathtt{R}_i = \mathtt{a}_i + \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasAddVIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)`

Calculates the interval vector `R` as the sum of the interval vector `A` and the vector `b`.

$$\mathtt{R}_i = \mathtt{A}_i + \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasAddVIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                 PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector `R` as the sum of the interval vectors `A` and `B`.

$$\mathtt{R}_i = \mathtt{A}_i + \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasSubVRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)`

Calculates the interval vector `R` which contains the true result of the difference between the vectors `a` and `b`.

$$\mathtt{R}_i = \mathtt{a}_i - \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasSubVRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)`

Calculates the interval vector `R` as the difference between the vector `a` and the interval vector `B`.

$$\mathtt{R}_i = \mathtt{a}_i - \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasSubVIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)
```

Calculates the interval vector R as the difference between the interval vector A and the vector b.

$$\mathtt{R}_i = \mathtt{A}_i - \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasSubVIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                 PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector R as the difference between the interval vectors A and B.

$$\mathtt{R}_i = \mathtt{A}_i - \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMulRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)
```

Calculates the interval vector R which contains the true result of the product between the scalar a and the vector b.

$$\mathtt{R}_i = \mathtt{a} \cdot \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMulRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector R which contains the true result of the product between the scalar a and the interval vector B.

$$\mathtt{R}_i = \mathtt{a} \cdot \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMulIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)
```

Calculates the interval vector R which contains the true result of the product between the scalar interval A and the vector b.

$$\mathtt{R}_i = \mathtt{A} \cdot \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMulIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector R which contains the true result of the product between the scalar interval A and the interval vector B.

$$\mathtt{R}_i = \mathtt{A} \cdot \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasMacRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)`

Adds to the interval vector `R` an interval vector which contains the true result of the product between the scalar `a` and the vector `b`.

$$\mathtt{R}_i = \mathtt{R}_i + \mathtt{a} \cdot \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasMacRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)`

Adds to the interval vector `R` an interval vector which contains the true result of the product between the scalar `a` and the interval vector `B`.

$$\mathtt{R}_i = \mathtt{R}_i + \mathtt{a} \cdot \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasMacIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)`

Adds to the interval vector `R` an interval vector which contains the true result of the product between the scalar interval `A` and the vector `b`.

$$\mathtt{R}_i = \mathtt{R}_i + \mathtt{A} \cdot \mathtt{b}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMacIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                PBIASINTERVAL pB, INT dim)
```

Adds to the interval vector `R` an interval vector which contains the true result of the product between the scalar interval `A` and the interval vector `B`.

$$\mathtt{R}_i = \mathtt{R}_i + \mathtt{A} \cdot \mathtt{B}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMacRVIs(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                 INT dim, INT Bstep)
```

Same as `BiasMacRVI` but with different stepsize for the components of `B`.

$$\mathtt{R}_i = \mathtt{R}_i + \mathtt{a} \cdot \mathtt{B}_{(i-1)\mathtt{Bstep}+1}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMacsRVIs(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                  INT dim, INT Rstep, INT Bstep)
```

Same as `BiasMacRVIs` but with different stepsize for the components of `R`.

$$\mathtt{R}_{(i-1)\mathtt{Rstep}+1} = \mathtt{R}_{(i-1)\mathtt{Rstep}+1} + \mathtt{a} \cdot \mathtt{B}_{(i-1)\mathtt{Bstep}+1}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasDivVRR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)
```

Calculates the interval vector R which contains the true result of the quotient of the vector a and the scalar b.

$$\mathtt{R}_i = \frac{\mathtt{a}_i}{\mathtt{b}}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasDivVRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector R which contains the true result of the quotient of the vector a and the scalar interval B.

$$\mathtt{R}_i = \frac{\mathtt{a}_i}{\mathtt{B}}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasDivVIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)
```

Calculates the interval vector R which contains the true result of the quotient of the interval vector A and the scalar b.

$$\mathtt{R}_i = \frac{\mathtt{A}_i}{\mathtt{b}}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasDivVII(PBIASINTERVAL pR, PBIASINTERVAL pA,
                PBIASINTERVAL pB, INT dim)
```

Calculates the interval vector R which contains the true result of the quotient of the interval vector A and the scalar interval B.

$$\mathtt{R}_i = \frac{\mathtt{A}_i}{\mathtt{B}}, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMacVRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)
```

Returns in the interval R an inclusion of the scalar product of the two vectors a and b.

```
VOID BiasMacVRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)
```

Returns in the interval R an inclusion of the scalar product of the vector a and the interval vector B.

```
VOID BiasMacVIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb, INT dim)
```

Returns in the interval `R` an inclusion of the scalar product of the interval vector `A` and the vector `b`.

```
VOID BiasMacVIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                 PBIASINTERVAL pB, INT dim)
```

Returns in the interval `R` an inclusion of the scalar product of the two interval vectors `A` and `B`.

```
VOID BiasInfV(PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector `r` containing the lower bounds of all components of the interval vector `A`.

$$\mathtt{r}_i = \inf \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasSupV(PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector `r` containing the upper bounds of all components of the interval vector `A`.

$$\mathtt{r}_i = \sup \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasNegV(PBIASINTERVAL pR, PBIASINTERVAL pA, INT dim)
```

Calculates the interval vector `R` containing the negative of the interval vector `A`.

$$\mathtt{r}_i = -\mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMidV(PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector `r` containing the midpoint of the interval vector `A`.

$$\mathtt{r}_i = \mathrm{mid}\, \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasMidRadV(PREAL pm, PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector m containing the midpoint of the interval vector A and the vector r containing the radius of A.

$$\begin{aligned} \mathtt{m}_i &= \operatorname{mid} \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim} \\ \mathtt{r}_i &= \operatorname{rad} \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim} \end{aligned}$$

```
VOID BiasDiamV(PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector r containing the diameters of all components of the interval vector A.

$$\mathtt{r}_i = \operatorname{diam} \mathtt{A}_i, \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasAbsV(PREAL pr, PBIASINTERVAL pA, INT dim)
```

Calculates the vector r containing the absolute values of all components of the interval vector A.

$$\mathtt{r}_i = |\mathtt{A}_i|, \quad i = 1, \ldots, \mathtt{dim}$$

```
INT BiasIntersectionV(PBIASINTERVAL pR, PBIASINTERVAL pA,
                      PBIASINTERVAL pB, INT dim)
```

Calculates the intersection vector R between the interval vectors A and B. If the intersection is empty, a value of 0 is returned and the contents of R are undefined. Otherwise, a value of 1 is returned.

```
VOID BiasHullVR(PBIASINTERVAL pR, PREAL pa, INT dim)
```

Converts the vector a to the point interval vector R.

$$\mathtt{R}_i = [\mathtt{a}_i, \mathtt{a}_i], \quad i = 1, \ldots, \mathtt{dim}$$

```
VOID BiasHullVRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT dim)
```

Initializes the interval vector R with the convex hull of the vectors a and b.

```
VOID BiasHullVRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB, INT dim)
```

Initializes the interval vector R with the convex hull of the vector a and the interval vector B.

```
VOID BiasHullVIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAK pb, INT dim)
```

Initializes the interval vector R with the convex hull of the interval vector A and the vector b.

```
VOID BiasHullVIVI(PBIASINTERVAL pR, PBIASINTERVAL pA,
                  PBIASINTERVAL pB, INT dim)
```

Initializes the interval vector R with the convex hull of the interval vectors A and B.

```
INT BiasInVR(PREAL pa, PBIASINTERVAL pB, INT dim)
```

Returns 1, if a $\in$ B, where a is a vector and B is an interval vector. Otherwise, 0 is returned.

```
INT BiasInVI(PBIASINTERVAL pA, PBIASINTERVAL pB, INT dim)
```

Returns 1, if A $\subseteq$ B, where A,B are interval vectors. Otherwise, 0 is returned.

```
INT BiasInInteriorVR(PREAL pa, PBIASINTERVAL pB, INT dim)
```

Returns 1, if a $\in$ Interior(B), where a is a vector and B is an interval vector. Otherwise, 0 is returned.

```
INT BiasInInteriorVI(PBIASINTERVAL pA, PBIASINTERVAL pB, INT dim)
```

Returns 1, if A $\subseteq$ Interior(B), where A,B are interval vectors. Otherwise, 0 is returned.

`VOID BiasSetToZeroV(PBIASINTERVAL pR, INT dim)`

Initializes the interval vector with zeroes, i.e.

$$\mathtt{R}_i = 0, \quad i = 1, \ldots, \mathtt{dim}$$

`VOID BiasSetToZeroVs(PBIASINTERVAL pR, INT dim, INT step)`

Same as before, but with the stepsize `step`.

$$\mathtt{R}_{(i-1)\mathtt{step}-1} = 0, \quad i = 1, \ldots, \mathtt{dim}$$

7 Matrix Operations — Bias2

Headerfile: Bias2.h

In all matrix routines the parameter `rows` denotes the number of rows in the matrix and the parameter `cols` denotes the number of columns. Throughout this description, elements are numbered beginning with the index 1. Matrix parameters are given as pointer to the elements where the pointer points to the first element. We denote a single matrix element by using indexing giving the row first.

`VOID BiasPredMR(PREAL pr, PREAL pa, INT rows, INT cols)`

The matrix `r` is set to the predecessor of the matrix `a`.

$$\mathtt{r}_{ij} = \mathrm{pred}(\mathtt{a}_{ij}), \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasSuccMR(PREAL pr, PREAL pa, INT rows, INT cols)`

The matrix `r` is set to the successor of the matrix `a`.

$$\mathtt{r}_{ij} = \mathrm{pred}(\mathtt{a}_{ij}), \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasPredMI(PBIASINTERVAL pR, PBIASINTERVAL pA, INT rows, INT cols)`

Each component of the resulting interval matrix `R` is set to the largest interval which is contained in the interior of the appropriate component of the interval matrix `A`.

```
VOID BiasSuccMI(PBIASINTERVAL pR, PBIASINTERVAL pA, INT rows, INT cols)
```

Each component of the resulting interval matrix R is set to the smallest interval where the appropriate component of the interval matrix A is contained in the interior.

```
VOID BiasAddMRMR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                 INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the sum of the matrices a and b.

$$\mathtt{R}_{ij} = \mathtt{a}_{ij} + \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasAddMRMI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the sum of the matrix a and the interval matrix B.

$$\mathtt{R}_{ij} = \mathtt{a}_{ij} + \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasAddMIMR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the sum of the interval matrix A and the matrix b.

$$\mathtt{R}_{ij} = \mathtt{A}_{ij} + \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasAddMIMI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the sum of the interval matrices A and B.

$$\mathtt{R}_{ij} = \mathtt{A}_{ij} + \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasSubMRMR(PBIASINTERVAL pR, PREAL pa, PREAL pb, INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the difference between the matrices a and b.

$$\mathtt{R}_{ij} = \mathtt{a}_{ij} - \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasSubMRMI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the difference between the matrix a and the interval matrix B.

$$\mathtt{R}_{ij} = \mathtt{a}_{ij} - \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasSubMIMR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the difference between the interval matrix A and the matrix b.

$$\mathtt{R}_{ij} = \mathtt{A}_{ij} - \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasSubMIMI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval matrix R as the difference between the interval matrices A and B.

$$\mathtt{R}_{ij} = \mathtt{A}_{ij} - \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasMulRMR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the product between the scalar a and the matrix b.

$$\mathtt{R}_{ij} = \mathtt{a} \cdot \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasMulRMI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the product between the scalar a and the interval matrix B.

$$\mathtt{R}_{ij} = \mathtt{a} \cdot \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasMulIMR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                INT rows, INT cols)
```

Calculates the interval matrix `R` which contains the true result of the product between the scalar interval `A` and the matrix `b`.

$$\mathtt{R}_{ij} = \mathtt{A} \cdot \mathtt{b}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasMulIMI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                INT rows, INT cols)
```

Calculates the interval matrix `R` which contains the true result of the product between the scalar interval `A` and the interval matrix `B`.

$$\mathtt{R}_{ij} = \mathtt{A} \cdot \mathtt{B}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasMulMRVR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                 INT rows, INT cols)
```

Calculates the interval vector `R` which contains the true result of the product between the matrix `a` and the vector `b`.

$$\mathtt{R}_i = \sum_{j=1}^{\mathtt{cols}} \mathtt{a}_{ij} \cdot \mathtt{b}_j, \quad i = 1, \ldots, \mathtt{rows}$$

```
VOID BiasMulMRVI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval vector `R` which contains the true result of the product between the matrix `a` and the interval vector `B`.

$$\mathtt{R}_i = \sum_{j=1}^{\mathtt{cols}} \mathtt{a}_{ij} \cdot \mathtt{B}_j, \quad i = 1, \ldots, \mathtt{rows}$$

```
VOID BiasMulMIVR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                 INT rows, INT cols)
```

Calculates the interval vector `R` which contains the true result of the product between the interval matrix `A` and the vector `b`.

$$\mathtt{R}_i = \sum_{j=1}^{\mathtt{cols}} \mathtt{A}_{ij} \cdot \mathtt{b}_j, \quad i = 1, \ldots, \mathtt{rows}$$

```
VOID BiasMulMIVI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                 INT rows, INT cols)
```

Calculates the interval vector R which contains the true result of the product between the interval matrix A and the interval vector B.

$$\mathtt{R}_i = \sum_{j=1}^{\mathtt{cols}} \mathtt{A}_{ij} \cdot \mathtt{B}_j, \quad i = 1, \ldots, \mathtt{rows}$$

```
VOID BiasMulMRMR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                 INT arows, INT acols, INT bcols)
```

Calculates the interval matrix R which contains the true result of the product between the matrices a and b.

$$\mathtt{R}_{ij} = \sum_{k=1}^{\mathtt{acols}} \mathtt{a}_{ik} \cdot \mathtt{b}_{kj}, \quad i = 1, \ldots, \mathtt{arows};\ j = 1, \ldots, \mathtt{bcols}$$

```
VOID BiasMulMRMI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                 INT arows, INT acols, INT Bcols)
```

Calculates the interval matrix R which contains the true result of the product between the matrix a and the interval matrix B.

$$\mathtt{R}_{ij} = \sum_{k=1}^{\mathtt{acols}} \mathtt{a}_{ik} \cdot \mathtt{B}_{kj}, \quad i = 1, \ldots, \mathtt{arows};\ j = 1, \ldots, \mathtt{Bcols}$$

```
VOID BiasMulMIMR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                 INT Arows, INT Acols, INT bcols)
```

Calculates the interval matrix R which contains the true result of the product between the interval matrix A and the matrix b.

$$\mathtt{R}_{ij} = \sum_{k=1}^{\mathtt{Acols}} \mathtt{A}_{ik} \cdot \mathtt{b}_{kj}, \quad i = 1, \ldots, \mathtt{Arows};\ j = 1, \ldots, \mathtt{bcols}$$

```
VOID BiasMulMIMI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                 INT Arows, INT Acols, INT Bcols)
```

Calculates the interval matrix R which contains the true result of the product between the interval matrices A and B.

$$\mathtt{R}_{ij} = \sum_{k=1}^{\mathtt{Acols}} \mathtt{A}_{ik} \cdot \mathtt{B}_{kj}, \quad i = 1, \ldots, \mathtt{Arows};\ j = 1, \ldots, \mathtt{Bcols}$$

```
VOID BiasDivMRR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the quotient of the matrix a and the scalar b.

$$\mathtt{R}_{ij} = \frac{\mathtt{a}_{ij}}{\mathtt{b}}, \quad i = 1,\ldots,\mathtt{rows};\ j = 1,\ldots,\mathtt{cols}$$

```
VOID BiasDivMRI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the quotient of the matrix a and the scalar interval B.

$$\mathtt{R}_{ij} = \frac{\mathtt{a}_{ij}}{\mathtt{B}}, \quad i = 1,\ldots,\mathtt{rows};\ j = 1,\ldots,\mathtt{cols}$$

```
VOID BiasDivMIR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAL pb,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the quotient of the interval matrix A and the scalar b.

$$\mathtt{R}_{ij} = \frac{\mathtt{A}_{ij}}{\mathtt{b}}, \quad i = 1,\ldots,\mathtt{rows};\ j = 1,\ldots,\mathtt{cols}$$

```
VOID BiasDivMII(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                INT rows, INT cols)
```

Calculates the interval matrix R which contains the true result of the quotient of the interval matrix A and the scalar interval B.

$$\mathtt{R}_{ij} = \frac{\mathtt{A}_{ij}}{\mathtt{B}}, \quad i = 1,\ldots,\mathtt{rows};\ j = 1,\ldots,\mathtt{cols}$$

```
VOID BiasInfM(PREAL pr, PBIASINTERVAL pA, INT rows, INT cols)
```

Calculates the matrix r containing the lower bounds of all components of the interval matrix A.

$$\mathtt{r}_{ij} = \inf \mathtt{A}_{ij}, \quad i = 1,\ldots,\mathtt{rows};\ j = 1,\ldots,\mathtt{cols}$$

`VOID BiasSupM(PREAL pr, PBIASINTERVAL pA, INT rows, INT cols)`

Calculates the matrix `r` containing the upper bounds of all components of the interval matrix `A`.

$$\mathtt{r}_{ij} = \sup \mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasNegM(PBIASINTERVAL pR, PBIASINTERVAL pA, INT rows, INT cols)`

Calculates the interval matrix `R` containing the negative of the interval matrix `A`.

$$\mathtt{r}_{ij} = -\mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasMidM(PREAL pr, PBIASINTERVAL pA, INT rows, INT cols)`

Calculates the matrix `r` containing the midpoint of the interval matrix `A`.

$$\mathtt{r}_{ij} = \mathrm{mid}\, \mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasMidRadM(PREAL pm, PREAL pr, PBIASINTERVAL pA,`
`INT rows, INT cols)`

Calculates the matrix `m` containing the midpoint of the interval matrix `A` and the matrix `r` containing the radius of `A`.

$$\begin{aligned} \mathtt{m}_{ij} &= \mathrm{mid}\, \mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols} \\ \mathtt{r}_{ij} &= \mathrm{rad}\, \mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols} \end{aligned}$$

`VOID BiasDiamM(PREAL pr, PBIASINTERVAL pA, INT rows, INT cols)`

Calculates the matrix `r` containing the diameters of all components of the interval matrix `A`.

$$\mathtt{r}_{ij} = \mathrm{diam}\, \mathtt{A}_{ij}, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

`VOID BiasAbsM(PREAL pr, PBIASINTERVAL pA, INT rows, INT cols)`

Calculates the matrix `r` containing the absolute values of all components of the interval matrix `A`.

$$\mathtt{r}_{ij} = |\mathtt{A}_{ij}|, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
INT BiasIntersectionM(PBIASINTERVAL pR, PBIASINTERVAL pA,
                      PBIASINTERVAL pB, INT rows, INT cols)
```

Calculates the intersection matrix R between the interval matrices A and B. If the intersection is empty, a value of 0 is returned and the contents of R are undefined. Otherwise, a value of 1 is returned.

```
VOID BiasHullMR(PBIASINTERVAL pR, PREAL pa, INT rows, INT cols)
```

Converts the matrix a to the point interval matrix R.

$$\mathtt{R}_{ij} = [\mathtt{a}_{ij}, \mathtt{a}_{ij}], \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

```
VOID BiasHullMRMR(PBIASINTERVAL pR, PREAL pa, PREAL pb,
                  INT rows, INT cols)
```

Initializes the interval matrix R with the convex hull of the matrices a and b.

```
VOID BiasHullMRMI(PBIASINTERVAL pR, PREAL pa, PBIASINTERVAL pB,
                  INT rows, INT cols)
```

Initializes the interval matrix R with the convex hull of the matrix a and the interval matrix B.

```
VOID BiasHullMIMR(PBIASINTERVAL pR, PBIASINTERVAL pA, PREAK pb,
                  INT rows, INT cols)
```

Initializes the interval matrix R with the convex hull of the interval matrix A and the matrix b.

```
VOID BiasHullMIMI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB,
                  INT rows, INT cols)
```

Initializes the interval matrix R with the convex hull of the interval matrices A and B.

`INT BiasInMR(PREAL pa, PBIASINTERVAL pB, INT rows, INT cols)`

Returns 1, if `a` $\in$ `B`, where `a` is a matrix and `B` is an interval matrix. Otherwise, 0 is returned.

`INT BiasInMI(PBIASINTERVAL pA, PBIASINTERVAL pB, INT rows, INT cols)`

Returns 1, if `A` $\subseteq$ `B`, where `A`,`B` are interval matrices. Otherwise, 0 is returned.

`INT BiasInInteriorMR(PREAL pa, PBIASINTERVAL pB, INT rows, INT cols)`

Returns 1, if `a` $\in$ Interior(`B`), where `a` is a matrix and `B` is an interval matrix. Otherwise, 0 is returned.

`INT BiasInInteriorMI(PBIASINTERVAL pA, PBIASINTERVAL pB, INT rows, INT cols)`

Returns 1, if `A` $\subseteq$ Interior(`B`), where `A`,`B` are interval matrices. Otherwise, 0 is returned.

`VOID BiasSetToZeroM(PBIASINTERVAL pR, INT rows, INT cols)`

Initializes the interval matrix with zeroes, i.e.

$$\mathtt{R}_{ij} = 0, \quad i = 1, \ldots, \mathtt{rows};\ j = 1, \ldots, \mathtt{cols}$$

8 Scalar Standard Functions — BiasF

Headerfile: BiasF.h

`VOID BiasFuncInit()`

Initializes all internal values, must be called prior any other BIAS standard function call.

`VOID BiasSin(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes `R` = sin(`A`)

`VOID BiasCos(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = cos(A)

`VOID BiasTan(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = tan(A)

`VOID BiasCot(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = cot(A)

`VOID BiasArcSin(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arcsin(A)

`VOID BiasArcCos(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arccos(A)

`VOID BiasArcTan(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arctan(A)

`VOID BiasArcCot(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arccot(A)

`VOID BiasSinh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = sinh(A)

`VOID BiasCosh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = cosh(A)

`VOID BiasTanh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = tanh(A)

`VOID BiasCoth(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = coth(A)

`VOID BiasArcSinh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arcsinh(A)

`VOID BiasArcCosh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arccosh(A)

`VOID BiasArcTanh(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arctanh(A)

`VOID BiasArcCoth(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = arccoth(A)

`VOID BiasExp(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = exp(A)

`VOID BiasLog(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = ln(A)

`VOID BiasLog10(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes R = $\log_{10}$(A)

`VOID BiasIAbs(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes $\mathtt{R} = \{\, |a| \mid a \in \mathtt{A} \,\}$

`VOID BiasSqr(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes $\mathtt{R} = \{\, a^2 \mid a \in \mathtt{A} \,\}$

`VOID BiasSqrt(PBIASINTERVAL pR, PBIASINTERVAL pA)`

Computes $\mathtt{R} = \sqrt{\mathtt{A}}$

`VOID BiasRoot(PBIASINTERVAL pR, PBIASINTERVAL pA, INT n)`

Computes $\mathtt{R} = \sqrt[\mathtt{n}]{\mathtt{A}}$

`VOID BiasPowerN(PBIASINTERVAL pR, PBIASINTERVAL pA, INT n)`

Computes $\mathtt{R} = \mathtt{A}^{\mathtt{n}}$

`VOID BiasPowerI(PBIASINTERVAL pR, PBIASINTERVAL pA, PBIASINTERVAL pB)`

Computes $\mathtt{R} = \mathtt{A}^{\mathtt{B}}$

In dieser Reihe sind bisher erschienen:

Bericht 90.1 (Juni 1990)
S. M. Rump: Approximate inverses of almost singular matrices still contain useful information

Bericht 90.2 (September 1990)
C. Jansson: A Posteriori Error Bounds for the Solution of Linear Systems with Simplicial Techniques

Bericht 90.3 (September 1990)
S. M. Rump: A class of arbitrarily ill-conditioned floating-point matrices

Bericht 90.4 (Oktober 1990)
C. Jansson: A Fast Direct Method for Computing Verified Inclusions

Bericht 90.5 (November 1990)
S. M. Rump: On the Solution of Interval Linear Systems

Bericht 91.1 (April 1991)
D. Husung: TPX Precompiler für Turbo Pascal

Bericht 91.2 (November 1991)
C. Jansson: A Global Minimization Method: The One-Dimensional Case

Bericht 91.3 (Dezember 1991)
S. M. Rump: Inclusion of the Solution for Large Linear Systems with M-Matrix

Bericht 92.1 (Januar 1992)
C. Jansson, O. Knüppel: A Global Minimization Method: The Multi-Dimensional Case

Bericht 92.2 (Februar 1992)
H.-P. Jüllig: Algorithmen mit Ergebnisverifikationen mit C++/2.0

Bericht 92.3 (Februar 1992)
S. M. Rump: Recent Results in Interval Mathematics

Bericht 92.4 (Februar 1992)
D. Husung: TPX Version 1.1 US Precompiler for Turbo Pascal 5.0

Bericht 92.5 (März 1992)
D. M. Claudio, S. M. Rump: Inclusion methods for real and complex functions in one variable

Bericht 92.6 (März 1992)
H.-P. Jüllig: BIBINS/2.0 C++-Bibliotheken für Vektoren und Matrizen über beliebigem skalaren Datentyp unter Berücksichtigung spezieller spärlicher Strukturen sowie Intervalldatentypen

Bericht 93.1 (Januar 1993)
S. M. Rump: Validated Solution of Large Linear Systems

Bericht 93.2 (Januar 1993)
D. Husung: TPX Version 1.1 US Precompiler for Turbo Pascal 5.0