Max-Planck-Institute for Gravitational Physics (Albert-Einstein-Institute)

UNIVERSITÀ DEGLI STUDI DI TRENTO

Arithmetic Operators in LTPDA

S2-AEI-TN-3059

prepared by	Martin Hewitson
checked by	
Issue	1
Revision	1
Status	Release
Number of pages	12
date of issue	October 1, 2012
approved by	

Part I.
Distribution List

Name	Company/ Institute
M. Hewitson	AEI Hannover

Contents

I. Distribution List **2**

1. Introduction **4**

2. Element-wise operators **4**
2.1. Container rules . . . 4
2.1.1. Rule 1 . . . 5
2.1.2. Rule 2 . . . 5
2.1.3. Rule 3 . . . 5
2.1.4. Rule 4 . . . 5
2.1.5. Rule 5 . . . 6
2.1.6. Rule 6 . . . 6
2.1.7. Rule 7 . . . 6
2.1.8. Rule 8 . . . 7
2.1.9. Rule 9 . . . 7
2.1.10. Rule 10 . . . 7
2.1.11. Rule 11 . . . 7
2.2. Data types . . . 8
2.3. Object names . . . 8
2.4. Units . . . 8
2.4.1. `plus`, `minus` . . . 8
2.4.2. `times`, `rdivide` . . . 9

3. Matrix operators **9**
3.1. Container rules . . . 9
3.1.1. Rule 1 . . . 9
3.1.2. Rule 2 . . . 10
3.1.3. Rule 3 . . . 10
3.1.4. Rule 4 . . . 10
3.1.5. Rule 5 . . . 10
3.1.6. Rule 6 . . . 11
3.1.7. Rule 7 . . . 11
3.1.8. Rule 8 . . . 11
3.1.9. Rule 9 . . . 11
3.1.10. Rule 10 . . . 11
3.2. Data types . . . 12
3.3. Object names . . . 12
3.4. Units . . . 12
3.4.1. `mtimes`, `mrdivide` . . . 12

List of Tables

1. Element-wise operators. . . . 4
2. Data type compatibility. . . . 8
3. Matrix operators. . . . 9
4. Data type compatibility. . . . 12

Operator	Symbol	description
`plus`	`+`	The addition operator.
`minus`	−	The subtraction operator.
`times`	`.*`	Element-wise multiplication operator.
`rdivide`	`./`	Element-wise division operator.

Table 1: Element-wise operators.

1. Introduction

This document aims to outline the desired behaviour for the various arithmetic operators for the AO class of LTPDA.

The aim is to describe how each operator should behave with regards:

- the inputs to the operator,
- the names of the input objects,
- the units of the input objects,
- and the allowed data types.

The data contained in the AO is always handled according to MATLAB's rules. Here we are only concerned with how the AO containers are handled. So, for example, if we want to do `v+b` where `v` is a vector of time-series AOs and `b` is a single time-series AO, we only discuss here how to handle the elements of `v`; after that, MATLAB handles how the data in `v(i)` and `b` are actually added together and will throw errors if MATLAB rules are violated. The rules are defined as a set of examples which should be exhaustive.

In this version of the document, data type `xyzdata` is not considered.

2. Element-wise operators

This section is concerned with those operators typically associated with operating element-wise on the data, for example, the `times` (`.*`) operator. The operators are detailed in Table 1.

2.1. Container rules

The following AO variables are assumed:

`a` a single AO

`b` a single AO

`c` a single AO

`V_N` a vector of N AOs

`U_N` a vector of N AOs

`M_NP` a Matrix of $N \times P$ AOs

`H_PQ` a Matrix of $P \times Q$ AOs

The following rules use `plus` as an example, but the use of the other operators is interchangeable.

2.1.1. Rule 1

```
o = plus(a,b)
```

Here we just add the data in `a` to the data in `b`, assuming the data conforms to MATLAB rules.

2.1.2. Rule 2

```
o = plus(V_N,a)
```

The output `o` will be equivalent to

```
[V(1)+a, V(2)+a, .. , V(N)+a]
```

so the size of `o` will be N elements.

2.1.3. Rule 3

```
o = plus(V_N,V_M)
```

will throw an error.

2.1.4. Rule 4

```
o = plus(V_N,U_N)
```

The output will be the vector:

```
[V(1)+U(1), V(2)+U(2), .. , V(N)+U(N)]
```

such that the size of ○ will be N elements.

2.1.5. Rule 5

```
○ = plus(M_NP,a)
```

will give an output

$$o = \begin{bmatrix} M(1,1)+a & M(1,2)+a & \cdots \\ M(2,1)+a & \ddots & \cdots \\ \vdots & \vdots & M(N,P)+a \end{bmatrix} \tag{1}$$

so ○ will be of size $N \times P$.

2.1.6. Rule 6

```
○ = plus(M_NP,V_N1)
```

will give an output

$$o = \begin{bmatrix} M(1,1)+V(1) & M(1,2)+V(1) & \cdots \\ M(2,1)+V(2) & \ddots & \cdots \\ \vdots & \vdots & M(N,P)+V(N) \end{bmatrix} \tag{2}$$

so ○ will be of size $N \times P$.

2.1.7. Rule 7

```
○ = plus(M_NP,V_1P)
```

will give an output

$$o = \begin{bmatrix} M(1,1)+V(1) & M(1,2)+V(2) & \cdots \\ M(2,1)+V(1) & \ddots & \cdots \\ \vdots & \vdots & M(N,P)+V(P) \end{bmatrix} \tag{3}$$

so o will be of size $N \times P$.

2.1.8. Rule 8

```
o = plus(M_NP,V_Q)
```

will give an error.

2.1.9. Rule 9

```
o = plus(M_NP,H_PQ)
```

will give an error.

2.1.10. Rule 10

```
o = plus(M_NP,H_NP)
```

will give an output

$$o = \begin{bmatrix} M(1,1)+H(1,1) & M(1,2)+H(1,2) & \cdots \\ M(2,1)+H(2,1) & \ddots & \cdots \\ \vdots & \vdots & M(N,P)+H(N,P) \end{bmatrix} \tag{4}$$

so o will be of size $N \times P$.

2.1.11. Rule 11

For more than two inputs, the result will be as if the commands are nested. For example:

```
o = plus(a,b,c)
```

will be handled the same as

```
o = plus(plus(a,b),c)
```

or for

```
o = plus(V_N,b,c)
```

	tsdata	fsdata	xydata	cdata
tsdata	y	n	y	y
fsdata	n	y	y	y
xydata	y	y	y	y
cdata	y	y	y	y

Table 2: Data type compatibility.

will be handled the same as

```
o = plus(plus(V_N,b),c)
```

At each stage, the above rules must apply.

2.2. Data types

The data compatibility matrix for these operators is given in Table 4.

2.3. Object names

The name of the output object will be based on the names of the input objects as

```
op = (in1.name op in2.name)
```

The name will be extended at each stage of a nested operation. For example, suppose we have three AOs, `a`, `b`, and `c`, each having a name the same as the variable name, then

```
o = plus(a,b,c)
```

will yield a name:

```
o.name = ((a+b)+c)
```

2.4. Units

2.4.1. `plus`, `minus`

For the addition operator, the units of the two inputs must be the same, or one (or both) must be empty (dimensionless). The output data will have units of the either input which is dimensioned.

Operator	Symbol	description
`mtimes`	*	The matrix multiplication operator.
`mrdivide`	/	The matrix division operator.

Table 3: Matrix operators.

2.4.2. `times`, `rdivide`

There are no restrictions for these operators. The units of the output data will be the product (or division) of the input units.

3. Matrix operators

This section is concerned with those operators typically associated with matrix manipulation. The operators are detailed in Table 3.

3.1. Container rules

The following AO variables are assumed:

`a` a single AO

`b` a single AO

`V_N` a vector of N AOs

`U_N` a vector of N AOs

`M_NP` a Matrix of $N \times P$ AOs

`H_PQ` a Matrix of $P \times Q$ AOs

The following rules use `mtimes` as an example, but the use of the other operators is interchangeable.

3.1.1. Rule 1

```
o = mtimes(a,b)
```

will yield an single output AO.

3.1.2. Rule 2

`o = mtimes(V_N1,a)` or `o = mtimes(V_1N,a)`

will behave the same as

```
o = times(V_N1,b)
```

for the AO containers, but the data will be processed by MATLAB according to the `mtimes` operator. So the output will be

```
o = [V(1)*b, V(2)*b, .. , V(N)*b]
```

3.1.3. Rule 3

```
o = mtimes(V_1N,U_N1)
```

will yield an output

```
o = [V(1)*U(1) + V(2)*U(2) + .. + V(N)*U(N)]
```

3.1.4. Rule 4

```
o = mtimes(V_1N,U_1N)
```

will throw an error.

3.1.5. Rule 5

```
o = mtimes(V_N1,U_1M)
```

will yield an output equivalent to

$$o = \begin{bmatrix} V(1)*U(1) & V(1)*U(2) & \cdots \\ V(2)*U(1) & \ddots & \cdots \\ \vdots & \vdots & V(N)*U(M) \end{bmatrix} \tag{5}$$

so of size $N \times M$.

3.1.6. Rule 6

```
o = mtimes(V_N1,U_N1)
```

will throw an error.

3.1.7. Rule 7

```
o = mtimes(M_NP,V_N1)
```

will throw an error.

3.1.8. Rule 8

```
o = mtimes(M_NP,V_1P)
```

will throw an error.

3.1.9. Rule 9

```
o = mtimes(M_NP,H_NP)
```

will throw an error.

3.1.10. Rule 10

```
o = mtimes(M_NP,H_PQ)
```

will yield an output

$$o = \begin{bmatrix} M(1,:) * H(:,1) & M(1,:) * H(:,2) & \cdots \\ M(2,:) * H(:,1) & \ddots & \cdots \\ \vdots & \vdots & M(N,:) * H(:,Q) \end{bmatrix} \tag{6}$$

so of size $N \times Q$.

	tsdata	fsdata	xydata	cdata
tsdata	y	n	y	y
fsdata	n	y	y	y
xydata	y	y	y	y
cdata	y	y	y	y

Table 4: Data type compatibility.

3.2. Data types

The data compatibility matrix for these operators is given in Table 4.

3.3. Object names

The name of the output object will be based on the names of the input objects as

```
out.name = [( in1.name op in2.name )]
```

The name will be extended at each stage of a nested operation. For example, suppose we have three AOs, `a`, `b`, and `c`, each having a name the same as the variable name, then

```
o = mtimes(a,b,c)
```

will yield a name:

```
o.name = ((a*b)*c)
```

3.4. Units

3.4.1. `mtimes`, `mrdivide`

There are no restrictions for these operators. The units of the output data will be the product (or division) of the input units.