Tensor and List Operations in CMAT©

Wolfgang M. Hartmann

July 2007

Contents

1 Introduction

1. Almost all arithmetic operations have been extended for tensors. For exceptions see below.
2. Almost all one and two argument standard functions have been extended for tensors. or exceptions see below.
3. Almost all index operations, including the subscript reduction operators, were extended to multidimensional arrays. For exceptions see below.
4. The `loc` and `replac` functions were extended to tensors.
5. The transpose operator `a‘` was extended to tensors.
6. The `all` and `any` functions were extended to tensors.
7. The `attrib`, `lstmem`, and `lstvar` function were extended for tensors and lists of objects.
8. The `obj2fil` and `fil2obj` functions were extended to tensors and lists of objects.
9. New specific functions for tensors were developed, see below.

Exceptions are:

1. The power operator `a ** b` works only for elementwise power, i.e. where `a` is scalar. However, the elementwise operator `.**` also works for tensors `a` and `b` with equal size.
2. The power function `pow(a,b)` works only for elementwise power, i.e. it will not work for scalar `b`.
3. The Kronecker product `a @ b` will not work for tensors when both `a` and `b` are scalars. If `a` or `b` is scalar the Krocker product reduces to the simple scalar times tensor product.
4. Tensor multiplication operator `a * b` assumes compatibility of `a` and `b`.
5. The concatenation operators are not yet implemented for tensors.
6. The index stacking operation `@` is not yet implemented for tensors.
7. The sort operations are not yet implemented for tensors.
8. No tensor extensions for the `flip`, `shape`, `spmat`, `rspfile`, and `wspfile` functions have been implemented until now.

2 Tensor Specification

2.1 Tensor Definition by Type Declaration

The type declarations `int`, `real`, `complex` and `char` can be used to allocate memory for tensors with constant values. If the declaration does not contain an explicit initialization by a constant or a scalar identifier, the entries are initialized to zero as it is for vectors and matrices.

```
print "Tensor definition by type declaration";
int A2[2,3,4]=9;
print "A2[2,3,4]=", A2;
```

Tensors are printed as lists of matrices. Here, there is tensor `a2` equivalent to a list two 3×4 matrices:

```
***************************
Tensor A2 with 3 Dimensions
***************************


                A2_1
                ****

    Dense Matrix (3 by 4)

  |       1      2      3      4
--------------------------------
1 |       9      9      9      9
2 |       9      9      9      9
3 |       9      9      9      9


                A2_2
                ****

    Dense Matrix (3 by 4)

  |       1      2      3      4
--------------------------------
1 |       9      9      9      9
2 |       9      9      9      9
3 |       9      9      9      9
```

```
real B1[3,4,5];
print "B1[3,4,5]=", B1;
```

```
**************************
Tensor B1 with 3 Dimensions
**************************
```

B1_1
****

Sparse Diagonal Matrix (4 by 5)

D	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

B1_2
****

Sparse Diagonal Matrix (4 by 5)

D	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

B1_3
****

Sparse Diagonal Matrix (4 by 5)

D	1	2	3	4	5

```
   ---------------------------------------
1 |       0      0      0      0      0
2 |       0      0      0      0      0
3 |       0      0      0      0      0
4 |       0      0      0      0      0
```

```
real B2[3,4,5] = 99.99;
print "B2[3,4,5]=", B2;
```

```
**************************
Tensor B2 with 3 Dimensions
**************************

                B2_1
                ****

           Dense Matrix (4 by 5)

  |          1          2          3          4          5
---------------------------------------------------------------
1 |   99.990000  99.990000  99.990000  99.990000  99.990000
2 |   99.990000  99.990000  99.990000  99.990000  99.990000
3 |   99.990000  99.990000  99.990000  99.990000  99.990000
4 |   99.990000  99.990000  99.990000  99.990000  99.990000

                B2_2
                ****

           Dense Matrix (4 by 5)

  |          1          2          3          4          5
---------------------------------------------------------------
1 |   99.990000  99.990000  99.990000  99.990000  99.990000
2 |   99.990000  99.990000  99.990000  99.990000  99.990000
3 |   99.990000  99.990000  99.990000  99.990000  99.990000
4 |   99.990000  99.990000  99.990000  99.990000  99.990000

                B2_3
                ****
```

```
                    Dense Matrix (4 by 5)

  |          1          2          3          4          5
-------------------------------------------------------------
1 |   99.990000  99.990000  99.990000  99.990000  99.990000
2 |   99.990000  99.990000  99.990000  99.990000  99.990000
3 |   99.990000  99.990000  99.990000  99.990000  99.990000
4 |   99.990000  99.990000  99.990000  99.990000  99.990000
```

The `dim` function is a straight forward extension of the `nrow` and `ncol` functions and can be used for obtaining the sizes of dimensions of vectors, matrices, tensors, and lists:

```
n2 = dim(B2);
print "Dimension B2=", n2;
```

```
Dimension B2=
   |   1   2   3
---------------
 1 |   3   4   5
```

```
n3 = dim(B2,3);
print "Number columns of B2=", n3;
```

```
Number columns of B2= 5
```

2.2 Function `t = const(`$nind <, val >$`)`

The function `t = const(`$nind <, val >$`)` is a tensor extension of the matrix function `m = cons(`$nr <, nc, val, type >$`)`. Assuming the tensor should have K dimensions the input vector `nind` should specify K integers as the upper index ranges of each dimension. The upper ranges of the K dimensions should only be the assumed lower bounds, using statements the ranges can still be increased later, by just defining tensor entries outside the specified ranges.

```
nind  = cons(3,1,4);    /* cons(nr,nc,val) */
```

```
atens = const(nind,.9);
print "Constant Tensor=", atens;
```

```
            ****************************
            Tensor atens with 3 Dimensions
            ****************************


                      atens_1
                      *******

                 Dense Matrix (4 by 4)

   |          1          2          3          4
---------------------------------------------------
1 |   0.9000000  0.9000000  0.9000000  0.9000000
2 |   0.9000000  0.9000000  0.9000000  0.9000000
3 |   0.9000000  0.9000000  0.9000000  0.9000000
4 |   0.9000000  0.9000000  0.9000000  0.9000000


                      atens_2
                      *******

                 Dense Matrix (4 by 4)

   |          1          2          3          4
---------------------------------------------------
1 |   0.9000000  0.9000000  0.9000000  0.9000000
2 |   0.9000000  0.9000000  0.9000000  0.9000000
3 |   0.9000000  0.9000000  0.9000000  0.9000000
4 |   0.9000000  0.9000000  0.9000000  0.9000000


                      atens_3
                      *******

                 Dense Matrix (4 by 4)

   |          1          2          3          4
---------------------------------------------------
1 |   0.9000000  0.9000000  0.9000000  0.9000000
2 |   0.9000000  0.9000000  0.9000000  0.9000000
3 |   0.9000000  0.9000000  0.9000000  0.9000000
4 |   0.9000000  0.9000000  0.9000000  0.9000000


                      atens_4
```

```
*******

Dense Matrix (4 by 4)

  |          1         2         3         4
-------------------------------------------------
1 |   0.9000000 0.9000000 0.9000000 0.9000000
2 |   0.9000000 0.9000000 0.9000000 0.9000000
3 |   0.9000000 0.9000000 0.9000000 0.9000000
4 |   0.9000000 0.9000000 0.9000000 0.9000000
```

2.3 Function `t = randt(`$nind <, type, ... >$`)`

The function `t = randt(`$nind <, type, ... >$`)` is a tensor extension of the matrix function `m = rand(`$< nr, nc, type, ... >$`)`. Assuming the tensor should have K dimensions the input vector `nind` should specify K integers as the upper index ranges of each dimension. The upper ranges of the K dimensions should only be the assumed lower bounds, using statements the ranges can still be increased later, by just defining tensor entries outside the specified ranges.

```
nind  = cons(3,1,4);    /* cons(nr,nc,val) */
btens = randt(nind,"duni");
print "Random Tensor=", btens;
```

```
*****************************
Tensor btens with 3 Dimensions
*****************************


                btens_1
                *******

            Dense Matrix (4 by 4)

  |          1         2         3         4
-------------------------------------------------
1 |   0.5227359 0.5820759 0.9495240 0.0589558
2 |   0.4789170 0.1752936 0.5748166 0.9191979
3 |   0.5654754 0.8211872 0.0233039 0.2533334
4 |   0.0391460 0.0291636 0.5537057 0.8072438


                btens_2
                *******
```

```
            Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.3755272  0.1559292  0.6907556  0.2778315
2 |   0.3591513  0.6645153  0.0457034  0.0892478
3 |   0.3212697  0.5177460  0.6297670  0.0041646
4 |   0.9522087  0.6289332  0.9003093  0.0292697


                  btens_3
                  *******

            Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.0436676  0.2648309  0.1362905  0.0320213
2 |   0.5454306  0.8316212  0.9688787  0.9546120
3 |   0.7059240  0.0738252  0.4689559  0.4186246
4 |   0.3660560  0.8317398  0.0331665  0.7558113


                  btens_4
                  *******

            Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.9461433  0.4727213  0.9416930  0.5162474
2 |   0.7402075  0.6150453  0.5592833  0.2504971
3 |   0.5451838  0.7267708  0.6763260  0.4590319
4 |   0.1381715  0.7493522  0.1386659  0.7232725
```

2.4 Names and Labels for Dimensions

b=dimname(a,nam)	nam=dimname(a)

Purpose: There are two versions of this function:

b=dimname(a,nam) two input arguments and no return: The returned object `b` is identical with input `a` except that the list of vectors `nam` of names is assigned to its dimensions. If the second argument is a missing value, any existing dimension names of `a` are removed.

nam=dimname(a) one input argument and one result: Returns a missing value if the dimensions of the object `a` have no names assigned, otherwise it returns the vector of dimension names of the object `a`.

Input: b=dimname(a,nam) The first argument is a tensor, matrix, or vector `a`. The second argument is a list of vectors `nam` of strings which should be used as names of the dimensios of `a`.

nam=dimname(a) The only input argument is a tensor, matrix, or vector `a`.

Output: b=name(a,nam) The returned object `b` is identical with input `a` except that the list of vectors `nam` of names is assigned to its dimensions.

nam=name(a) Returns a missing value if the dimensions of object `a` have no names assigned, otherwise it returns the list of vectors of dimension names of the object `a`.

Restrictions: 1. The second input argument must be a list of vectors of strings that does not contain any numeric or missing values.

2. The number of strings in the list of vectors in second argument must match the size of dimensions of the first input argument.

Relationships: dimlabel(), rname(), cname(), clabel(), rlabel().

Examples: 1. This easy examples illustrates how to create a list of variable names, how to attach it to dhe dimensions of a tensor, and how to move it from a tensor back into a list of string vectors.

```
srand(1);
nind  = [ 2 3 4 ];
atens = randt(nind,"duni");

list tnam[3];
tnam[1] = [ "time1" "time2" ];
tnam[2] = [ "row1" "row2" "row3" ];
tnam[3] = [ "col1" "col2" "col3" "col4" ];
print "List tnam=",tnam;


List tnam=
                        ************************
                        List tnam with 3 Entries
                        ************************


                               tnam[1]
                               *******
```

```
                    Dense Row Vector (ncol=2)

                       R |        1       2
                              time1   time2


                             tnam[2]
                             *******

                    Dense Row Vector (ncol=3)

                        R |       1      2      3
                               row1   row2   row3


                             tnam[3]
                             *******

                    Dense Row Vector (ncol=4)

                   R |       1      2      3      4
                          col1   col2   col3   col4


  ctens = dimname(atens,tnam);
  print "Ctens=",ctens;


Ctens=
                  ****************************
                  Tensor ctens with 3 Dimensions
                  ****************************


                             ctens_1
                             *******

                        Dense Matrix (3 by 4)

               |        col1       col2       col3       col4
            ---------------------------------------------------
          row1 |   0.1849626  0.9700887  0.3998243  0.2593986
          row2 |   0.9216026  0.9692773  0.5429792  0.5316917
          row3 |   0.0497940  0.0665666  0.8193186  0.5238705


                             ctens_2
```

```
                        *******

                    Dense Matrix (3 by 4)

             |        col1       col2       col3       col4
          ---------------------------------------------------
     row1 |   0.8533943  0.0671846  0.9570239  0.2971940
     row2 |   0.2726118  0.6899296  0.9767649  0.2265075
     row3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

```
/* get the list back: should be the same as tnam */
vnam = dimname(ctens);
print "Original names=",vnam;
```

```
Original names=
                    ************************
                    List vnam with 3 Entries
                    ************************


                           vnam[1]
                           *******

                    Dense Column Vector (nrow=2)

                        C |        1       2
                                time1   time2


                           vnam[2]
                           *******

                    Dense Column Vector (nrow=3)

                      C |       1      2      3
                             row1   row2   row3


                           vnam[3]
                           *******

                    Dense Column Vector (nrow=4)

                   C |       1      2      3      4
                          col1   col2   col3   col4
```

b=dimlabel(a,lab)	lab=dimlabel(a)

Purpose: There are two versions of this function:

b=dimlabel(a,lab) two input arguments and no return: The returned object `b` is identical with input `a` except that a list of vectors `lab` with label strings is assigned to its dimensions. If the second argument is a missing value, any existing dimension labels of `a` are removed.

lab=dimlabel(a) one input argument and one result: Returns a missing value if the object `a` has no dimension labels assigned, otherwise it returns a list of vectors with dimension labels of the object `a`.

Input: b=dimlabel(a,lab) The first argument is a tensor, matrix or vector `a`. The second argument is a list of vectors `lab` of strings which should be used as labels of the dimensions of `a`.

lab=dimlabel(a) The only input argument is a tensor, matrix, or vector `a`.

Output: b=dimlabel(a,lab) The returned object `b` is identical with the input object `a` except that the list of vectors `lab` of labels is assigned to its dimensions.

lab=dimlabel(a) Returns a missing value if the dimensions of the object `a` have no labels assigned, otherwise it returns the list of vectors of dimension labels of the object `a`.

Restrictions:
1. The second input argument must be a vector of strings that does not contain any numeric or missing values.
2. The number of strings in the second argument must match the number of columns of the first input argument.

Relationships: dimname(), rname(), cname(), rlabel(), clabel()

Examples: See the `dimname` function above.

3 Simple Tensor Operations

3.1 Permute Tensor Dimensions: `TENPERM`

The function `to = tenperm(ti,ordr)` can be used for permuting the dimensions of a tensor.

```
ordr = [ 3 2 1 ];
ctens = tenperm(btens,ordr);
print "Permuted Indices=",ctens;
```

Permuted Indices=

****
Tensor ctens with 3 Dimensions
****

ctens_1
****

Dense Matrix (4 by 4)

	1	2	3	4
1	0.5227359	0.3755272	0.0436676	0.9461433
2	0.4789170	0.3591513	0.5454306	0.7402075
3	0.5654754	0.3212697	0.7059240	0.5451838
4	0.0391460	0.9522087	0.3660560	0.1381715

ctens_2
****

Dense Matrix (4 by 4)

	1	2	3	4
1	0.5820759	0.1559292	0.2648309	0.4727213
2	0.1752936	0.6645153	0.8316212	0.6150453
3	0.8211872	0.5177460	0.0738252	0.7267708
4	0.0291636	0.6289332	0.8317398	0.7493522

ctens_3
****

Dense Matrix (4 by 4)

	1	2	3	4
1	0.9495240	0.6907556	0.1362905	0.9416930
2	0.5748166	0.0457034	0.9688787	0.5592833
3	0.0233039	0.6297670	0.4689559	0.6763260
4	0.5537057	0.9003093	0.0331665	0.1386659

ctens_4
****

Dense Matrix (4 by 4)

```
  |          1          2          3          4
-------------------------------------------------
1 |   0.0589558  0.2778315  0.0320213  0.5162474
2 |   0.9191979  0.0892478  0.9546120  0.2504971
3 |   0.2533334  0.0041646  0.4186246  0.4590319
4 |   0.8072438  0.0292697  0.7558113  0.7232725
```

3.2 Move Tensor to List of Matrices: TEN2MAT

The function `mlst = ten2mat(tens<,ordr>);` can be used to move a tensor to a list of matrices.

```
nind  = cons(3,1,4);    /* cons(nr,nc,val) */
btens = randt(nind,"duni");
print "Random Tensor=", btens;
```

```
Random Tensor=

                ****************************
                Tensor btens with 3 Dimensions
                ****************************

                          btens_1
                          *******

                     Dense Matrix (4 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.1615326  0.7821955  0.8198760  0.1816163
2 |   0.3839244  0.3459368  0.2220190  0.6055288
3 |   0.5259491  0.0853359  0.2895431  0.1075468
4 |   0.7749259  0.6610987  0.1972289  0.2294105


                          btens_2
                          *******

                     Dense Matrix (4 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.3936414  0.5776659  0.2598749  0.1643267
```

```
2 |   0.7545403  0.8493966  0.3872302  0.9103450
3 |   0.0151857  0.9133448  0.2087599  0.4406090
4 |   0.0063326  0.3148612  0.7310842  0.1049515


                    btens_3
                    *******

              Dense Matrix (4 by 4)

  |          1          2          3          4
-----------------------------------------------
1 |   0.7236164  0.4824903  0.8112518  0.4846813
2 |   0.2917406  0.4730618  0.2494313  0.7311393
3 |   0.2807263  0.2316055  0.8921028  0.2311551
4 |   0.4825279  0.7883688  0.8275581  0.9260677


                    btens_4
                    *******

              Dense Matrix (4 by 4)

  |          1          2          3          4
-----------------------------------------------
1 |   0.0076426  0.3004769  0.6560177  0.0625198
2 |   0.9401413  0.6350539  0.2925110  0.6809179
3 |   0.8969033  0.4506592  0.8169410  0.9702098
4 |   0.1582244  0.5267348  0.0188727  0.5607032


  matlst = ten2mat(btens); /* lstmem(1); */
  print "List of Matrices=", matlst;


List of Matrices=

                   *************************
                   List matlst with 4 Entries
                   *************************


                         matlst[1]
                         *********

                   Dense Matrix (4 by 4)

             |          1          2          3          4
```

```
-------------------------------------------------
1 |   0.1615326  0.7821955  0.8198760  0.1816163
2 |   0.3839244  0.3459368  0.2220190  0.6055288
3 |   0.5259491  0.0853359  0.2895431  0.1075468
4 |   0.7749259  0.6610987  0.1972289  0.2294105
```

```
                matlst[2]
                *********

           Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.3936414  0.5776659  0.2598749  0.1643267
2 |   0.7545403  0.8493966  0.3872302  0.9103450
3 |   0.0151857  0.9133448  0.2087599  0.4406090
4 |   0.0063326  0.3148612  0.7310842  0.1049515
```

```
                matlst[3]
                *********

           Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.7236164  0.4824903  0.8112518  0.4846813
2 |   0.2917406  0.4730618  0.2494313  0.7311393
3 |   0.2807263  0.2316055  0.8921028  0.2311551
4 |   0.4825279  0.7883688  0.8275581  0.9260677
```

```
                matlst[4]
                *********

           Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.0076426  0.3004769  0.6560177  0.0625198
2 |   0.9401413  0.6350539  0.2925110  0.6809179
3 |   0.8969033  0.4506592  0.8169410  0.9702098
4 |   0.1582244  0.5267348  0.0188727  0.5607032
```

3.3 Move List of Matrices to Tensor: MAT2TEN

The function `tens = mat2ten(mlst,nind<,ordr>);` can be used to move a list of matrices to a tensor.

```
nind = [ 4 4 4 ];
dtens = mat2ten(matlst,nind);
print "Tensor made from matrix list=", dtens;
```

```
Tensor made from matrix list=

                  ****************************
                  Tensor dtens with 3 Dimensions
                  ****************************


                           dtens_1
                           *******

                      Dense Matrix (4 by 4)

    |           1          2          3          4
----------------------------------------------------
1 |   0.1615326  0.7821955  0.8198760  0.1816163
2 |   0.3839244  0.3459368  0.2220190  0.6055288
3 |   0.5259491  0.0853359  0.2895431  0.1075468
4 |   0.7749259  0.6610987  0.1972289  0.2294105


                           dtens_2
                           *******

                      Dense Matrix (4 by 4)

    |           1          2          3          4
----------------------------------------------------
1 |   0.3936414  0.5776659  0.2598749  0.1643267
2 |   0.7545403  0.8493966  0.3872302  0.9103450
3 |   0.0151857  0.9133448  0.2087599  0.4406090
4 |   0.0063326  0.3148612  0.7310842  0.1049515


                           dtens_3
                           *******

                      Dense Matrix (4 by 4)
```

```
  |          1          2          3          4
-------------------------------------------------
1 |   0.7236164  0.4824903  0.8112518  0.4846813
2 |   0.2917406  0.4730618  0.2494313  0.7311393
3 |   0.2807263  0.2316055  0.8921028  0.2311551
4 |   0.4825279  0.7883688  0.8275581  0.9260677


                    dtens_4
                    *******

             Dense Matrix (4 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.0076426  0.3004769  0.6560177  0.0625198
2 |   0.9401413  0.6350539  0.2925110  0.6809179
3 |   0.8969033  0.4506592  0.8169410  0.9702098
4 |   0.1582244  0.5267348  0.0188727  0.5607032
```

3.4 Move Tensor to single Vector: TEN2VEC

vec = ten2vec(tens¡,ordr¿)

Purpose: The `ten2vec` function moves all the entries of a tensor `tens` to a vector `vec`.

Input: **tens** Specifies an existing tensor whose entries should be moved to a vector.

ordr This is an optional permutation vector specifying the order of the dimensions how the entries of the tensor are moved into the vector.

Output: The single return is a vector containing all entries of the tensor. The order of the dimensions maybe specified by the `ordr` vector.

Restrictions: 1. Tensors with string data are not tested yet.

2. The size of the `ordr` vector must be the same as the dimensionality of the tensor, its entries must define a permutation. The integer values depend on the INDBASE option which is by default equal to one.

Relationships: vec2ten(), ten2mat(), mat2ten

Examples: 1. Dense Tensor:

```
  srand(1);
  nind  = [ 2 3 4 ];
  atens = randt(nind,"duni");


  print " * Dense Tensor *";
  print "Atens=", atens;
```

```
Atens=
                  ******************************
                  Tensor atens with 3 Dimensions
                  ******************************


                           atens_1
                           *******

                       Dense Matrix (3 by 4)

             |           1          2          3          4
           ---------------------------------------------------
           1 |   0.1849626  0.9700887  0.3998243  0.2593986
           2 |   0.9216026  0.9692773  0.5429792  0.5316917
           3 |   0.0497940  0.0665666  0.8193186  0.5238705


                           atens_2
                           *******

                       Dense Matrix (3 by 4)

             |           1          2          3          4
           ---------------------------------------------------
           1 |   0.8533943  0.0671846  0.9570239  0.2971940
           2 |   0.2726118  0.6899296  0.9767649  0.2265075
           3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

```
  cvec = ten2vec(atens);
  print "cvec=", cvec;
```

```
cvec=
   |         1
--------------
 1 |   0.18496
 2 |   0.97009
```

```
 3 |   0.39982
 4 |   0.25940
 5 |   0.92160
 6 |   0.96928
 7 |   0.54298
 8 |   0.53169
 9 |   0.04979
10 |   0.06657
11 |   0.81932
12 |   0.52387
13 |   0.85339
14 |   0.06718
15 |   0.95702
16 |   0.29719
17 |   0.27261
18 |   0.68993
19 |   0.97676
20 |   0.22651
21 |   0.68824
22 |   0.41276
23 |   0.55855
24 |   0.28723
```

```
dtens = vec2ten(cvec,nind);
print "dtens should be equal to atens=", dtens;
```

```
                ****************************
                Tensor dtens with 3 Dimensions
                ****************************


                          dtens_1
                          *******

                     Dense Matrix (3 by 4)

              |          1          2          3          4
            ------------------------------------------------
            1 |  0.1849626  0.9700887  0.3998243  0.2593986
            2 |  0.9216026  0.9692773  0.5429792  0.5316917
            3 |  0.0497940  0.0665666  0.8193186  0.5238705


                          dtens_2
                          *******
```

```
                    Dense Matrix (3 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.8533943  0.0671846  0.9570239  0.2971940
2 |   0.2726118  0.6899296  0.9767649  0.2265075
3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

2. Sparse Tensor:

```
print " * Sparse Tensor *";
btens = atens > .8;
print "Btens=",btens;
```

```
                  ******************************
                  Tensor btens with 3 Dimensions
                  ******************************


                            btens_1
                            *******

                        Sparse Matrix (3 by 4)

                    |       1      2      3      4
                  ---------------------------------
                  1 |       0      1      0      0
                  2 |       1      1      0      0
                  3 |       0      0      1      0


                            btens_2
                            *******

                Sparse Upper Triangular Matrix (3 by 4)

                    U |       1      2      3      4
                  ---------------------------------
                  1 |       1      0      1      0
                  2 |       0      0      1      0
                  3 |       0      0      0      0
```

```
cvec = ten2vec(btens);
print "cvec=", cvec;
```

```
                              cvec
                              ****

                    Sparse Column Vector cvec

      C |       2      5      6     11     13     15     19
                1      1      1      1      1      1      1
```

```
dtens = vec2ten(cvec,nind);
print "dtens should be equal to btens=", dtens;
```

```
                  ****************************
                  Tensor dtens with 3 Dimensions
                  ****************************


                            dtens_1
                            *******

                       Sparse Matrix (3 by 4)

                    |       1      2      3      4
                  --------------------------------
                  1 |       0      1      0      0
                  2 |       1      1      0      0
                  3 |       0      0      1      0


                            dtens_2
                            *******

                 Sparse Upper Triangular Matrix (3 by 4)

                  U |       1      2      3      4
                  --------------------------------
                  1 |       1      0      1      0
                  2 |       0      0      1      0
                  3 |       0      0      0      0
```

3.5 Move single Vector to Tensor: VEC2TEN

tens = vec2ten(vec,dims)

Purpose: The `vec2ten` function moves all the entries of a a vector `vec` to tensor `tens`.

Input: **vec** Specifies the vector whose entries should be moved to the tensor.

dims Specifies the size of the dimensions and should be a vector of integers. The product of these integers should correspond to the size of the input vector `vec`.

Output: Output is a tensor with dimensions `dims` containing the values of the vector. If the vector has less entries than the tensor size `dims`, the remaining entries of the tensor are set to zero.

Restrictions: 1. Tensors with string data are not tested yet.

2. The size of the vector `vec` and the product of the integers in `dims` should be about the same. If there are less entries in the vector than the size of the tensor defined by `dims` the rest of the tensor is filled with zeros.

Relationships: ten2vec(), ten2mat(), mat2ten

Examples: 1. Dense Vector:

```
srand(1);
nind  = [ 2 3 4 ];
atens = randt(nind,"duni");


print " * Dense Tensor *";
print "Atens=", atens;
```

```
Atens=
                    ****************************
                    Tensor atens with 3 Dimensions
                    ****************************


                              atens_1
                              *******

                         Dense Matrix (3 by 4)

      |           1          2          3          4
------------------------------------------------------
1 |   0.1849626  0.9700887  0.3998243  0.2593986
2 |   0.9216026  0.9692773  0.5429792  0.5316917
3 |   0.0497940  0.0665666  0.8193186  0.5238705
```

```
                          atens_2
                          *******

                    Dense Matrix (3 by 4)

             |           1          2          3          4
          ---------------------------------------------------
          1 |   0.8533943  0.0671846  0.9570239  0.2971940
          2 |   0.2726118  0.6899296  0.9767649  0.2265075
          3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

```
cvec = ten2vec(atens);
print "cvec=", cvec;
```

```
cvec=
   |         1
--------------
 1 |   0.18496
 2 |   0.97009
 3 |   0.39982
 4 |   0.25940
 5 |   0.92160
 6 |   0.96928
 7 |   0.54298
 8 |   0.53169
 9 |   0.04979
10 |   0.06657
11 |   0.81932
12 |   0.52387
13 |   0.85339
14 |   0.06718
15 |   0.95702
16 |   0.29719
17 |   0.27261
18 |   0.68993
19 |   0.97676
20 |   0.22651
21 |   0.68824
22 |   0.41276
23 |   0.55855
24 |   0.28723
```

```
dtens = vec2ten(cvec,nind);
print "dtens should be equal to atens=", dtens;
```

```
*****************************
Tensor dtens with 3 Dimensions
*****************************


                dtens_1
                *******

            Dense Matrix (3 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.1849626  0.9700887  0.3998243  0.2593986
2 |   0.9216026  0.9692773  0.5429792  0.5316917
3 |   0.0497940  0.0665666  0.8193186  0.5238705


                dtens_2
                *******

            Dense Matrix (3 by 4)

  |          1          2          3          4
-------------------------------------------------
1 |   0.8533943  0.0671846  0.9570239  0.2971940
2 |   0.2726118  0.6899296  0.9767649  0.2265075
3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

2. Sparse Vector:

```
print " * Sparse Tensor *";
btens = atens > .8;
print "Btens=",btens;
```

```
*****************************
Tensor btens with 3 Dimensions
*****************************


                btens_1
                *******

            Sparse Matrix (3 by 4)
```

```
  |       1      2      3      4
--------------------------------
1 |       0      1      0      0
2 |       1      1      0      0
3 |       0      0      1      0
```

```
btens_2
*******
```

```
Sparse Upper Triangular Matrix (3 by 4)
```

```
U |       1      2      3      4
--------------------------------
1 |       1      0      1      0
2 |       0      0      1      0
3 |       0      0      0      0
```

```
cvec = ten2vec(btens);
print "cvec=", cvec;
```

```
cvec
****
```

```
Sparse Column Vector cvec
```

```
C |       2      5      6     11     13     15     19
          1      1      1      1      1      1      1
```

```
dtens = vec2ten(cvec,nind);
print "dtens should be equal to btens=", dtens;
```

```
*****************************
Tensor dtens with 3 Dimensions
*****************************
```

```
dtens_1
*******
```

```
Sparse Matrix (3 by 4)
```

```
  |       1      2      3      4
--------------------------------
```

```
  1 |       0      1      0      0
  2 |       1      1      0      0
  3 |       0      0      1      0


                    dtens_2
                    *******

          Sparse Upper Triangular Matrix (3 by 4)

     U |       1      2      3      4
     --------------------------------
     1 |       1      0      1      0
     2 |       0      0      1      0
     3 |       0      0      0      0
```

3.6 Add Scalar to Tensors

The addition and subtraction between scalars and tensors is done in a straight forward way:

```
csca = 3.;
ctens = atens + csca;
print "Add scalar to tensor=",ctens;
```

```
Add scalar to tensor=

                    ****************************
                    Tensor ctens with 3 Dimensions
                    ****************************

                              ctens_1
                              *******

                         Dense Matrix (4 by 4)

       |          1          2          3          4
     ------------------------------------------------
     1 |   3.9000000  3.9000000  3.9000000  3.9000000
     2 |   3.9000000  3.9000000  3.9000000  3.9000000
     3 |   3.9000000  3.9000000  3.9000000  3.9000000
     4 |   3.9000000  3.9000000  3.9000000  3.9000000
```

```
                    ctens_2
                    *******

             Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   3.9000000  3.9000000  3.9000000  3.9000000
2 |   3.9000000  3.9000000  3.9000000  3.9000000
3 |   3.9000000  3.9000000  3.9000000  3.9000000
4 |   3.9000000  3.9000000  3.9000000  3.9000000


                    ctens_3
                    *******

             Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   3.9000000  3.9000000  3.9000000  3.9000000
2 |   3.9000000  3.9000000  3.9000000  3.9000000
3 |   3.9000000  3.9000000  3.9000000  3.9000000
4 |   3.9000000  3.9000000  3.9000000  3.9000000


                    ctens_4
                    *******

             Dense Matrix (4 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   3.9000000  3.9000000  3.9000000  3.9000000
2 |   3.9000000  3.9000000  3.9000000  3.9000000
3 |   3.9000000  3.9000000  3.9000000  3.9000000
4 |   3.9000000  3.9000000  3.9000000  3.9000000
```

3.7 Apply `lstmem` and `lstvar`

Test the functions `lstmem` (list memory usage) and `lstvar` (list allocated variables) with detail output for Lists and Tensors:

```
lstmem(1);
lstvar(1);
```

Detail Table of Memory Usage
****

Block		N		Begin	Length	Content
1	0	1	U	1	148	atens
1	1	2	U	149	512	ctens_2
1		3	F	661	340	
Used		2			660	
Free		1			340	
Block		N		Begin	Length	Content
2	5	4	U	1001	512	atens_2
2	6	5	U	1513	148	btens
2	3	6	U	1661	156	nind
2	2	7	U	1817	148	ctens
2		8	F	1965	36	
Used		4			964	
Free		1			36	
Block		N		Begin	Length	Content
3	7	9	U	2001	512	btens_2
3		10	F	2513	488	
Used		1			512	
Free		1			488	
Block		N		Begin	Length	Content
4		11	F	3001	1000	
Used		0			0	
Free		1			1000	
Block		N		Begin	Length	Content
5		12	F	4001	1000	
Used		0			0	
Free		1			1000	
Block		N		Begin	Length	Content
6		13	F	5001	1000	

```
 Used            0                     0
 Free            1                  1000
-------------------------------------------------------------


   Summary Table of Memory Usage
   *****************************

 Memory Blocks Allocated:             6
 Number of Bytes Allocated:        6000
 Number of Bytes Used:             2136
 Number of Bytes Free:             3864

   List of 7 Largest Allocations
-------------------------------------------------------------
   Rank     Block     Begin    Length   Content
-------------------------------------------------------------
      1         1       149       512   ctens_2
      2         2      1001       512   atens_2
      3         3      2001       512   btens_2
      4         2      1661       156   nind
      5         1         1       148   atens
      6         2      1513       148   btens
      7         2      1817       148   ctens
-------------------------------------------------------------


         Local Variables
         ***************

-------------------------------------------------------
        Name        Nest  Type           Content
-------------------------------------------------------
     1  ipr            1  INT                  1
     2  nind           1  VECTOR             INT
     3  atens          1  TENSOR            REAL
     4  btens          1  TENSOR            REAL
     5  csca           1  REAL       3.000000000
     6  ctens          1  TENSOR            REAL
-------------------------------------------------------
```

3.8 Add and Subtract Tensors

For `c = a + b` or `c = a - b` either both operands `a` and `b` can be tensors of equal size or one of the operands can be scalar. The addition or subtraction operation is performed elementwise and the result is always a tensor `c`.

```
  ctens = atens + btens;
  print "Add Tensor=", ctens;
```

```
Add Tensor=

                ****************************
                Tensor ctens with 3 Dimensions
                ****************************

                          ctens_1
                          *******

                     Dense Matrix (4 by 4)

  |          1          2          3          4
--------------------------------------------------
1 |   1.0615326  1.6821955  1.7198760  1.0816163
2 |   1.2839244  1.2459368  1.1220190  1.5055288
3 |   1.4259491  0.9853359  1.1895431  1.0075468
4 |   1.6749259  1.5610987  1.0972289  1.1294105


                          ctens_2
                          *******

                     Dense Matrix (4 by 4)

  |          1          2          3          4
--------------------------------------------------
1 |   1.2936414  1.4776659  1.1598749  1.0643267
2 |   1.6545403  1.7493966  1.2872302  1.8103450
3 |   0.9151857  1.8133448  1.1087599  1.3406090
4 |   0.9063326  1.2148612  1.6310842  1.0049515


                          ctens_3
                          *******

                     Dense Matrix (4 by 4)

  |          1          2          3          4
--------------------------------------------------
1 |   1.6236164  1.3824903  1.7112518  1.3846813
2 |   1.1917406  1.3730618  1.1494313  1.6311393
3 |   1.1807263  1.1316055  1.7921028  1.1311551
4 |   1.3825279  1.6883688  1.7275581  1.8260677
```

```
                        ctens_4
                        *******

                    Dense Matrix (4 by 4)

      |          1          2          3          4
------------------------------------------------------
    1 |   0.9076426  1.2004769  1.5560177  0.9625198
    2 |   1.8401413  1.5350539  1.1925110  1.5809179
    3 |   1.7969033  1.3506592  1.7169410  1.8702098
    4 |   1.0582244  1.4267348  0.9188727  1.4607032


  ctens = atens - btens;
  print "Subtract Tensor=", ctens;


Subtract Tensor=

                  ****************************
                  Tensor ctens with 3 Dimensions
                  ****************************

                        ctens_1
                        *******

                    Dense Matrix (4 by 4)

      |          1          2          3          4
------------------------------------------------------
    1 |   0.7384674  0.1178045  0.0801240  0.7183837
    2 |   0.5160756  0.5540632  0.6779810  0.2944712
    3 |   0.3740509  0.8146641  0.6104569  0.7924532
    4 |   0.1250741  0.2389013  0.7027711  0.6705895


                        ctens_2
                        *******

                    Dense Matrix (4 by 4)

      |          1          2          3          4
------------------------------------------------------
    1 |   0.5063586  0.3223341  0.6401251  0.7356733
    2 |   0.1454597  0.0506034  0.5127698 -0.0103450
    3 |   0.8848143 -0.0133448  0.6912401  0.4593910
    4 |   0.8936674  0.5851388  0.1689158  0.7950485
```

```
                    ctens_3
                    *******

             Dense Matrix (4 by 4)

  |           1          2          3          4
-----------------------------------------------
1 |   0.1763836  0.4175097  0.0887482  0.4153187
2 |   0.6082594  0.4269382  0.6505687  0.1688607
3 |   0.6192737  0.6683945  0.0078972  0.6688449
4 |   0.4174721  0.1116312  0.0724419 -0.0260677


                    ctens_4
                    *******

             Dense Matrix (4 by 4)

  |           1          2          3          4
-----------------------------------------------
1 |   0.8923574  0.5995231  0.2439823  0.8374802
2 |  -0.0401413  0.2649461  0.6074890  0.2190821
3 |   0.0030967  0.4493408  0.0830590 -0.0702098
4 |   0.7417756  0.3732652  0.8811273  0.3392968
```

4 Tensor Multiplication

4.1 Multiply Tensor with Scalar

For `c = a * b` and where one of the operands is scalar, the operation is performed elementwise and the result is always a tensor `c`.

```
scal = 3.;
etens = scal * btens;
print "Scalar times Tensor=", etens;
```

```
Scalar times Tensor=

                ******************************
                Tensor etens with 3 Dimensions
                ******************************

                          etens_1
```

****

Dense Matrix (4 by 4)

| 1 | 2 | 3 | 4
---|---|---|---|---
1 | 0.4845978 | 2.3465866 | 2.4596280 | 0.5448488
2 | 1.1517732 | 1.0378103 | 0.6660569 | 1.8165865
3 | 1.5778473 | 0.2560078 | 0.8686292 | 0.3226405
4 | 2.3247776 | 1.9832961 | 0.5916867 | 0.6882315

etens_2
****

Dense Matrix (4 by 4)

| 1 | 2 | 3 | 4
---|---|---|---|---
1 | 1.1809243 | 1.7329978 | 0.7796247 | 0.4929800
2 | 2.2636208 | 2.5481899 | 1.1616907 | 2.7310350
3 | 0.0455571 | 2.7400343 | 0.6262798 | 1.3218269
4 | 0.0189979 | 0.9445837 | 2.1932526 | 0.3148546

etens_3
****

Dense Matrix (4 by 4)

| 1 | 2 | 3 | 4
---|---|---|---|---
1 | 2.1708491 | 1.4474710 | 2.4337553 | 1.4540438
2 | 0.8752218 | 1.4191854 | 0.7482938 | 2.1934179
3 | 0.8421788 | 0.6948166 | 2.6763084 | 0.6934653
4 | 1.4475837 | 2.3651064 | 2.4826743 | 2.7782030

etens_4
****

Dense Matrix (4 by 4)

| 1 | 2 | 3 | 4
---|---|---|---|---
1 | 0.0229277 | 0.9014306 | 1.9680532 | 0.1875594
2 | 2.8204239 | 1.9051616 | 0.8775331 | 2.0427537
3 | 2.6907099 | 1.3519777 | 2.4508230 | 2.9106295
4 | 0.4746732 | 1.5802045 | 0.0566180 | 1.6821096

4.2 Compatibility of Tensor Multiplication with $*$

Compatibility issues of tensor multiplication with $*$:

1. $\mathbf{C} = \mathbf{A} * \mathbf{B}$, `A` is tensor, $\mathbf{B}$ is matrix or vector:

   If a tensor $\mathbf{A}$ is multiplied with a matrix or vector $\mathbf{B}$, then the size of the last dimension of the tensor $\mathbf{A}$ must be the same as the number of rows of $\mathbf{B}$. When $\mathbf{B}$ is vector, then $d_C = d_A - 1$. When $\mathbf{B}$ is matrix, then $d_C = d_A$.

2. $\mathbf{C} = \mathbf{A} * \mathbf{B}$, `B` is tensor, $\mathbf{A}$ is matrix or vector:

   If a matrix or vector $\mathbf{A}$ is multiplied with a tensor $\mathbf{B}$, then the number of columns of $\mathbf{A}$ must be the same as the size of the first dimension of the tensor $\mathbf{B}$. When $\mathbf{A}$ is vector, then $d_C = d_B - 1$. When $\mathbf{A}$ is matrix, then $d_C = d_B$.

3. $\mathbf{C} = \mathbf{A} * \mathbf{B}$, where $\mathbf{A}$, `B` are tensors:

   For $\mathbf{C} = \mathbf{A} * \mathbf{B}$, where both, $\mathbf{A}$ and $\mathbf{B}$ are tensors the dimensionsionality d_C of $\mathbf{C}$ is:

   - $d_C = 0$ i.e. $\mathbf{C}$ is scalar: when $\mathbf{A}$ and $\mathbf{B}$ have the same size.
   - $d_C = d_1 + d_2$: if s is the number of first dimensions with equal size, then $d_1 = d_A - s$ and $d_2 = d_B - s$.

   The functions `tentvec`, `tentmat`, and `tentten` can be used for more specific forms of tensor multiplication.

4.3 Multiply Tensor with (List of) Vector(s)

```
tens|scal|vec = tentvec(tens,vec<,n>);
```

The `tentvec` function implements three different ways to multiply a tensor `aten` with one or more vectors `bvec`. Assuming tensor`aten` has dimensionality d with sizes $(m_1, \ldots, m_d)$.

1. n mode multiplication of tensor `aten` with one vector `bvec`: Assuming `bvec` has size m.

   - optional third argument `n` is not specified: `n` is assumed equal to d and the vector size m must be the same as m_d
   - optional third argument `n` is specified: the integer `n` must refer to a dimension of `aten` and the vector size m must be the same as m_n

The result of the multiplication is a tensor with dimension $d-1$.

2. multiplication of tensor `aten` with a list of d vectors `bvec`: Any specification of n is ignored. The sizes of the d vectors in the list must agree with the tensor sizes $(m_1, \ldots, m_d)$. The result of the multiplication is a scalar.

3. multiplication of tensor `aten` with a list of $d-1$ vectors `bvec`:

   - optional third argument `n` is not specified: `n` is assumed equal to d.
   - optional third argument `n` is specified: the integer `n` must refer to a dimension of `aten`.

   The sizes of the $d-1$ vectors in the list must agree with the tensor sizes $(m_1, \ldots, m_{n-1}, m_{n+1}, \ldots, m_d)$. The result of the multiplication is a vector with size m_n.

1. Multiply tensor with a single vector:

```
srand(1);
nind  = [ 2 3 4 ];
atens = randt(nind,"duni");
```

```
Random Tensor Atens=

                  ****************************
                  Tensor atens with 3 Dimensions
                  ****************************


                           atens_1
                           *******

                       Dense Matrix (3 by 4)

              |           1          2          3          4
            ------------------------------------------------
            1 |   0.1849626  0.9700887  0.3998243  0.2593986
            2 |   0.9216026  0.9692773  0.5429792  0.5316917
            3 |   0.0497940  0.0665666  0.8193186  0.5238705


                           atens_2
                           *******

                       Dense Matrix (3 by 4)
```

```
  |          1          2          3          4
-------------------------------------------------
1 |   0.8533943  0.0671846  0.9570239  0.2971940
2 |   0.2726118  0.6899296  0.9767649  0.2265075
3 |   0.6882366  0.4127639  0.5585541  0.2872256
```

```
cvec = [ 10. 10. ];
cmat = tentvec(atens,cvec,1);
print "Tensor * Vector (ten1vec) Cmat=",cmat;
```

```
Tensor * Vector (ten1vec) Cmat=
   |         1         2         3         4
-------------------------------------------
 1 |    10.384    10.373    13.568    5.5659
 2 |    11.942    16.592    15.197    7.5820
 3 |    7.3803    4.7933    13.779    8.1110
```

2. Multiply tensor with a list of d vectors:

```
list blist[3];
blist[1] = [ 10. 10. ];
blist[2] = [ 20. 20. 20. ];
blist[3] = [ 30. 30. 30. 30. ];
cscal = tentvec(atens,blist);
print "Tensor * VectorList (tennvec) Cscal=",cscal;
```

```
Tensor * VectorList (tennvec) Cscal= 75160.59
```

3. Multiply tensor with a list of $d-1$ vectors:

```
list blist[2];
blist[1] = [ 10. 10. ];
blist[2] = [ 30. 30. 30. 30. ];
cvec = tentvec(atens,blist,2);
print "Tensor * VectorList (tenmvec) Cvec=",cvec;
```

```
Tensor * VectorList (tenmvec) Cvec=
   |         1
```

```
--------------
 1 |    1196.7
 2 |    1539.4
 3 |    1021.9
```

4.4 Multiply Tensor with (List of) Matrix(ces)

```
tens = tentmat(tens,mat<,n>);
```

The `tentmat` function implements three different ways to multiply a tensor with one or more matrices `bmat`. Assuming tensor`aten` has dimensionality d with sizes $(m_1, \ldots, m_d)$.

1. n mode multiplication of tensor `aten` with one matrix `bmat`: Assuming `bmat` has size $mr \times mc$.

   - optional third argument `n` is not specified: `n` is assumed equal to d and the row number mr must be the same as m_d
   - optional third argument `n` is specified: the integer `n` must refer to a dimension of `aten` and the row number mr must be the same as m_n

   The result of the multiplication is a tensor with dimension d where dimension n has now size m_c, like the number of columns of matrix `bmat`.

2. multiplication of tensor `aten` with a list of d matrices `bmat`: Any specification of n is ignored. The row numbers of the d matrices in the list must agree with the tensor sizes $(m_1, \ldots, m_d)$. The result of the multiplication is a tensor with dimensionality d and sizes like the columns numbers of the d matrices.

3. multiplication of tensor `aten` with a list of $d-1$ matrices `bmat`:

   - optional third argument `n` is not specified: `n` is assumed equal to d.
   - optional third argument `n` is specified: the integer `n` must refer to a dimension of `aten`.

   The row numbers of the $d-1$ matrices in the list must agree with the tensor sizes $(m_1, \ldots, m_{n-1}, m_{n+1}, \ldots, m_d)$. The result of the multiplication is a tensor with d dimensions with sizes corresponding to the column sizes of the $d-1$ matrices except of that of dimension n which is of the same size m_n as that tensor `aten`.

1. Multiply tensor with a single matrix:

```
srand(1);
nind  = [ 2 3 4 ];
atens = randt(nind,"duni");
bmat = [ 10. 10. 10. ,
         10. 10. 10. ];
cten1 = tentmat(atens,bmat,1);
print "Tensor * Matrix (ten1mat) Cten1=",cten1;
```

```
Tensor * Matrix (ten1mat) Cten1=

                ******************************
                Tensor cten1 with 3 Dimensions
                ******************************


                          cten1_1
                          *******

                     Dense Matrix (3 by 4)

              |           1          2          3          4
             ---------------------------------------------------
             1 |   10.383569  10.372733  13.568482  5.5659261
             2 |   11.942144  16.592070  15.197440  7.5819924
             3 |   7.3803058  4.7933042  13.778727  8.1109613


                          cten1_2
                          *******

                     Dense Matrix (3 by 4)

              |           1          2          3          4
             ---------------------------------------------------
             1 |   10.383569  10.372733  13.568482  5.5659261
             2 |   11.942144  16.592070  15.197440  7.5819924
             3 |   7.3803058  4.7933042  13.778727  8.1109613


                          cten1_3
                          *******

                     Dense Matrix (3 by 4)

              |           1          2          3          4
```

```
-----------------------------------------------
1 |   10.383569  10.372733  13.568482  5.5659261
2 |   11.942144  16.592070  15.197440  7.5819924
3 |   7.3803058  4.7933042  13.778727  8.1109613
```

2. Multiply tensor with a list of d matrices:

```
list blist[3];
blist[1] = [ 10. 10. 10. ,
             10. 10. 10. ];
blist[2] = [ 20. 20. 20. ,
             20. 20. 20. ,
             20. 20. 20. ];
blist[3] = [ 30. 30. 30. ,
             30. 30. 30. ,
             30. 30. 30. ,
             30. 30. 30. ];
ctens = tentmat(atens,blist);
print "Tensor * MatrixList (tennmat) Ctens=",ctens;
```

```
Tensor * MatrixList (tennmat) Ctens=

                ******************************
                Tensor ctens with 3 Dimensions
                ******************************

                           ctens_1
                           *******

                Dense Symmetric Matrix (3 by 3)

               S |           1          2          3
               -------------------------------------
               1 |   75160.592
               2 |   75160.592  75160.592
               3 |   75160.592  75160.592  75160.592


                           ctens_2
                           *******

                Dense Symmetric Matrix (3 by 3)
```

```
S |           1          2          3
-------------------------------------
1 |   75160.592
2 |   75160.592  75160.592
3 |   75160.592  75160.592  75160.592


             ctens_3
             *******

 Dense Symmetric Matrix (3 by 3)

S |           1          2          3
-------------------------------------
1 |   75160.592
2 |   75160.592  75160.592
3 |   75160.592  75160.592  75160.592
```

3. Multiply tensor with a list of $d-1$ matrices:

```
list blist[2];
blist[1] = [ 10. 10. 10. ,
             10. 10. 10. ];
blist[2] = [ 30. 30. 30. ,
             30. 30. 30. ,
             30. 30. 30. ,
             30. 30. 30. ];
cmat = tentmat(atens,blist,2);
print "Tensor * MatrixList (tenmmat) Cmat=",cmat;


Tensor * MatrixList (tenmmat) Cmat=

                ****************************
                Tensor cmat with 3 Dimensions
                ****************************

                           cmat_1
                           ******

                    Dense Matrix (3 by 3)

                |          1          2          3
               -------------------------------------
```

```
  1 |   1196.7213  1196.7213  1196.7213
  2 |   1539.4094  1539.4094  1539.4094
  3 |   1021.8989  1021.8989  1021.8989


                cmat_2
                ******

      Dense Matrix (3 by 3)

    |           1          2          3
----------------------------------------
  1 |   1196.7213  1196.7213  1196.7213
  2 |   1539.4094  1539.4094  1539.4094
  3 |   1021.8989  1021.8989  1021.8989


                cmat_3
                ******

      Dense Matrix (3 by 3)

    |           1          2          3
----------------------------------------
  1 |   1196.7213  1196.7213  1196.7213
  2 |   1539.4094  1539.4094  1539.4094
  3 |   1021.8989  1021.8989  1021.8989
```

4.5 Multiply Tensor with Tensor

```
tens|scal = tentten(aten,bten<,n>);
```

The `tentten` function implements two different ways to multiply a tensor with a tensor.

1. Both tensors `aten` and `bten` have same dimension and size and optional third argument `n` is not be specified or specified equal to dimensionality. Result of multiplication is scalar which is the dot product of corresponding entries of `aten` and `bten`.

2. Both tensors `aten` and `bten` may have different size: If the third argument `n` is not specified, `n` is obtained as the number of first dimensions of `aten` and `bten` which have the same size. If `n` is specified at least the first n dimensions of `aten` and `bten` must have same size. Assuming ad and bd

are the dimensionalities of `aten` and `bten`, then the sum of the dimensions with unequal size $(ad - n) + (bd - n)$ is the dimensionality of the result tensor, matrix, vector or scalar.

1. Tensors of equal size: Result is scalar:

```
srand(1);
nind  = [ 2 3 4 ];
atens = randt(nind,"duni");
btens = randt(nind,"duni");
cscal = tentten(atens,btens);
print "Tensor * Tensor (tennten) Cscal=",cscal;
```

```
                ****************************
                Tensor atens with 3 Dimensions
                ****************************


                          atens_1
                          *******

                     Dense Matrix (3 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.1849626  0.9700887  0.3998243  0.2593986
2 |   0.9216026  0.9692773  0.5429792  0.5316917
3 |   0.0497940  0.0665666  0.8193186  0.5238705


                          atens_2
                          *******

                     Dense Matrix (3 by 4)

  |           1          2          3          4
-------------------------------------------------
1 |   0.8533943  0.0671846  0.9570239  0.2971940
2 |   0.2726118  0.6899296  0.9767649  0.2265075
3 |   0.6882366  0.4127639  0.5585541  0.2872256


Tensor * Tensor (tennten) Cscal= 6.7656
```

2. Tensors of unequal size: Result has dimension of sum of unequal dimensions:

```
mind  = [ 2 3 3 ];
btens = randt(mind,"duni");
cmat = tentten(atens,btens,2);
print "Tensor * Tensor (tenmten) Cmat=",cmat;
```

```
Tensor * Tensor (tenmten) Cmat=
   |         1         2         3
---------------------------------
 1 |    10.705    8.1924    4.1397
 2 |    11.444    8.7583    4.4257
 3 |    15.331    11.733    5.9289
 4 |    7.6608    5.8628    2.9626
```

5 Additional Remarks

6 The Bibliography

References

[1] Bader, B. W. & Kolda, T. G. (200x), "A Preliminary Report on the Development of MATLAB Tensor Classes for Fast Algorithm Prototyping", Technical Report SAND2004-3487, Sandia National Laboratory.

[2] Bader, B. W. & Kolda, T. G. (200x), "Algorithm xxx: MATLAB tensor classes for fast algorithm prototyping", *ACM TOMS*, .

[3] Golub, G., & Van Loan, C.F. (1989), *Matrix Computations*, John Hopkins University Press, 2nd ed., Baltimore, MD.

[4] Hartmann, W. (2015), "Data Objects in CMAT", Technical Report CMAT, Heidelberg.

[5] Hartmann, W. (2006), "CMAT - Tutorial", Technical Report CMAT, Heidelberg.

[6] Kolda, T. G. (2006), "Multilinear operators for higher-order decompositions", Technical Report SAND2006-2081, Sandia National Laboratory.