Multidimensional arrays with Julia

Lana Periša, FESB, University of Split

Split Applied Mathematics Day 2018

15.6.2018.

Why Julia?

- free and open source language developed on MIT
- familiar mathematical notation like Matlab
- powerful for linear algebra as Matlab

- fast as C
- usable for general programming as Python
- dynamic as Ruby
- easy for statistics as R
- natural for string processing as Perl
- good at gluing programs together as the shell

Tensors

$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ N-dimensional array of real numbers

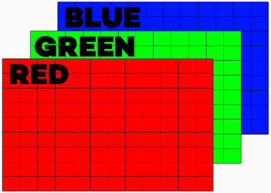
$\rightarrow$ tensor of order N

$\rightarrow$ has N modes

$\rightarrow$ elements: $x_{i_1 i_2 \ldots i_N}, \quad i_n \in \{1, 2, \ldots, I_n\}, \quad n = 1, 2, \ldots, N$

Tensor of order 1 (vector):	`X[i]`
Tensor of order 2 (matrix):	`X[i,j]`
Tensor of order 3:	`X[i,j,k]`
$\vdots$	
Tensor of order N:	`X[i1,i2,...,iN]`

Example 1: Digital images



- I_1, I_2, I_3 ... pixels
- I_4 ... person
- I_5 ... angle
- I_6 ... illumination

$$\rightarrow \quad \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5 \times I_6}$$

Example 2: Multivariate functions

- $u : [0,1]^N \to \mathbb{R}$
- domain discretization:

$$0 \leq \xi_1^{(n)} < \xi_2^{(n)} < \cdots < \xi_{I_n}^{(n)} \leq 1, \quad n = 1, 2, \ldots, N$$

- evaluate the function on the grid:

$$x_{i_1 i_2 \ldots i_N} = u(\xi_{i_1}^{(1)}, \xi_{i_2}^{(2)}, \ldots, \xi_{i_N}^{(N)}), \quad i_n = 1, 2, \ldots, I_n$$

$\rightarrow \quad \mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$

Multidimensional algorithms

- algorithms working with tensors

Multidimensional algorithms

- algorithms working with tensors

Example: Iterate over elements

```
function iterate(X::Array{T,N}) where T<:Number,N
 I=size(X)
 for i1=1:I[1]
   for i2=1:I[2]
     ...
      for iN=1:I[N]
        X[i1,i2,...,iN]
      end
     ...
   end
 end
end
```

$\rightarrow$ switch to linear indexing:

```
X[i1,i2,...,iN]  ↔  X[i]=vec(X)[i]
```

Example: $\mathfrak{X} \in \mathbb{R}^{5\times 4}$

$$X = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}$$

$\rightarrow$ switch to linear indexing:

```
X[i1,i2,...,iN]  ↔  X[i]=vec(X)[i]
```

Example: $\mathcal{X} \in \mathbb{R}^{5\times 4}$

$$X = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}$$

```
X[8]=X[3,2]
X[19]=X[4,4]
```

Example: $\mathcal{X} \in \mathbb{R}^{5\times 4\times 3}$

`X[:,:,1]`

$$\begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}$$

`X[:,:,2]`

$$\begin{bmatrix} 21 & 26 & 31 & 36 \\ 22 & 27 & 32 & 37 \\ 23 & 28 & 33 & 38 \\ 24 & 29 & 34 & 39 \\ 25 & 30 & 35 & 40 \end{bmatrix}$$

`X[:,:,3]`

$$\begin{bmatrix} 41 & 46 & 51 & 56 \\ 42 & 47 & 52 & 57 \\ 43 & 48 & 53 & 58 \\ 44 & 49 & 54 & 59 \\ 45 & 50 & 55 & 60 \end{bmatrix}$$

Example: $\mathcal{X} \in \mathbb{R}^{5\times4\times3}$

$$\underset{\texttt{X[:,:,1]}}{\begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix}} \quad \underset{\texttt{X[:,:,2]}}{\begin{bmatrix} 21 & 26 & 31 & 36 \\ 22 & 27 & 32 & 37 \\ 23 & 28 & 33 & 38 \\ 24 & 29 & 34 & 39 \\ 25 & 30 & 35 & 40 \end{bmatrix}} \quad \underset{\texttt{X[:,:,3]}}{\begin{bmatrix} 41 & 46 & 51 & 56 \\ 42 & 47 & 52 & 57 \\ 43 & 48 & 53 & 58 \\ 44 & 49 & 54 & 59 \\ 45 & 50 & 55 & 60 \end{bmatrix}}$$

```
X[26]=X[1,2,2]
X[49]=X[4,2,3]
```

$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N$

- $(i_1, i_2, \ldots, i_N) \quad \rightarrow \quad i$

  `sub2ind(size(X),(i1,i2,...,iN)...)`

- $i \quad \rightarrow \quad (i_1, i_2, \ldots, i_N)$

  `ind2sub(size(X),i)`

$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N$

- $(i_1, i_2, \ldots, i_N) \quad \rightarrow \quad i$

  `sub2ind(size(X),(i1,i2,...,iN)...)`

- $i \quad \rightarrow \quad (i_1, i_2, \ldots, i_N)$

  `ind2sub(size(X),i)`

Example: Iterate over elements

```
function iterate(X::Array{T}) where T<:Number
 for i=1:length(X)  % length(X)=numel(X) from MATLAB
    X[i]
 end
end
```

Type of problems we deal with

$\mathfrak{X} \ldots I_1 \times I_2 \times \cdots \times I_n \quad \rightarrow \quad I_1 I_2 \cdots I_N$ elements

Type of problems we deal with

$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_n \quad \rightarrow \quad I_1 I_2 \cdots I_N$ elements

Example:

`X=rand(50,50,50,50,50)` $\rightarrow$ 50^5 elements

$\rightarrow$ memory problems

Type of problems we deal with

$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_n \quad \rightarrow \quad I_1 I_2 \cdots I_N$ elements

Example:

`X=rand(50,50,50,50,50)` $\rightarrow$ 50^5 elements

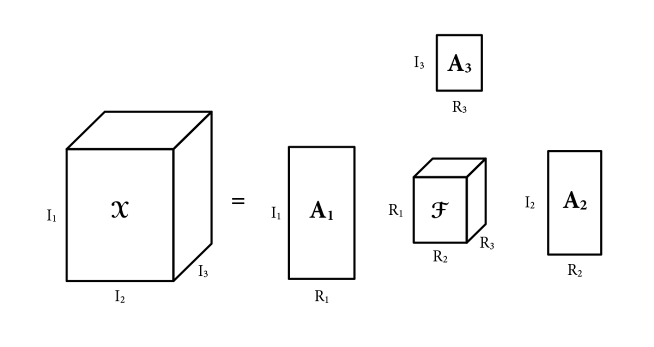
$\rightarrow$ memory problems

Compressed formats:

- CP format
- Tucker format
- Hierarchical Tucker format
- Tensor Train format

Tensors in Tucker format

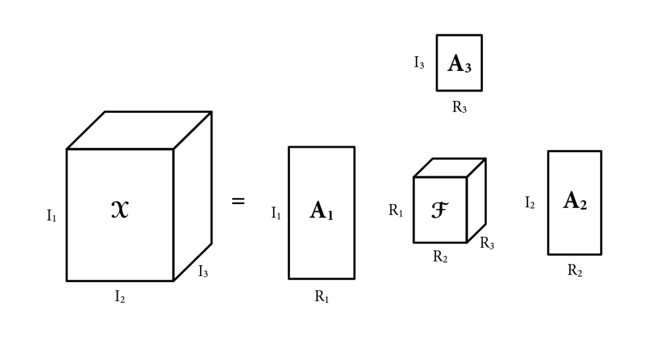
- for $N = 3$:



Storage: $I_1 I_2 I_3 \quad \leftrightarrow \quad R_1 R_2 R_3 + I_1 R_1 + I_2 R_2 + I_3 R_3$

Tensors in Tucker format

- for $N = 3$:



Storage: $\quad I_1 I_2 I_3 \quad \leftrightarrow \quad R_1 R_2 R_3 + I_1 R_1 + I_2 R_2 + I_3 R_3$

e.g. $\quad 10^6 = 100^3 \quad \leftrightarrow \quad 10^3 + 3 \times 100 \times 10 = 4 \times 10^3$

Tensors in Tucker format

If $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be represented as

$$\mathcal{X} = \mathcal{F} \times_1 \mathsf{A}_1 \times_2 \mathsf{A}_2 \times_3 \cdots \times_N \mathsf{A}_N,$$

we say it is in Tucker format.

- $\mathcal{F} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ $\quad \rightarrow \quad$ core tensor $\quad$ ($R_n \leq I_n, \forall n$)
- $\mathsf{A}_n \in \mathbb{R}^{I_n \times R_n}$, $n = 1, \ldots, N$ $\quad \rightarrow \quad$ factor matrices
  (usually orthonormal)

Storage:

$$I_1 I_2 \cdots I_N \quad \leftrightarrow \quad R_1 R_2 \cdots R_N + \sum_{n=1}^{N} I_n R_n$$

Operations with tensors in Tucker format

- ADDITION

$$\mathcal{X} = \mathcal{F} \times_1 \mathsf{A}_1 \times_2 \mathsf{A}_2 \times_3 \mathsf{A}_3, \quad \mathcal{F} \in \mathbb{R}^{P_1 \times P_2 \times P_3},$$
$$\mathcal{Y} = \mathcal{G} \times_1 \mathsf{B}_1 \times_2 \mathsf{B}_2 \times_3 \mathsf{B}_3, \quad \mathcal{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3},$$

$$\mathcal{X} + \mathcal{Y} = (\mathcal{F} \oplus \mathcal{G}) \times_1 \left[\mathsf{A}^{(1)} \ \ \mathsf{B}^{(1)} \right] \times_2 \left[\mathsf{A}^{(2)} \ \ \mathsf{B}^{(2)} \right] \times_3 \left[\mathsf{A}^{(3)} \ \ \mathsf{B}^{(3)} \right]$$

Operations with tensors in Tucker format

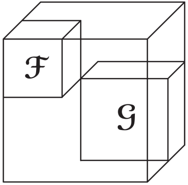
- ADDITION

$$\mathcal{X} = \mathcal{F} \times_1 A_1 \times_2 A_2 \times_3 A_3, \quad \mathcal{F} \in \mathbb{R}^{P_1 \times P_2 \times P_3},$$
$$\mathcal{Y} = \mathcal{G} \times_1 B_1 \times_2 B_2 \times_3 B_3, \quad \mathcal{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3},$$

$$\mathcal{X} + \mathcal{Y} = (\mathcal{F} \oplus \mathcal{G}) \times_1 \left[A^{(1)} \;\; B^{(1)} \right] \times_2 \left[A^{(2)} \;\; B^{(2)} \right] \times_3 \left[A^{(3)} \;\; B^{(3)} \right]$$

$$\downarrow$$

$$diag\,(\mathcal{F}, \mathcal{G}) =$$



$$\in \mathbb{R}^{(P_1+Q_1)\times(P_2+Q_2)\times(P_3+Q_3)}$$

Operations with tensors in Tucker format

- ELEMENTWISE PRODUCT

$$\mathcal{X} = \mathcal{F} \times_1 \mathsf{A}_1 \times_2 \mathsf{A}_2 \times_3 \mathsf{A}_3, \quad \mathcal{F} \in \mathbb{R}^{P_1 \times P_2 \times P_3},$$
$$\mathcal{Y} = \mathcal{G} \times_1 \mathsf{B}_1 \times_2 \mathsf{B}_2 \times_3 \mathsf{B}_3, \quad \mathcal{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3},$$

$$\mathcal{X} * \mathcal{Y} = (\mathcal{F} \otimes \mathcal{G}) \times_1 \left(\mathsf{A}_1 \odot^T \mathsf{B}_1\right) \times_2 \left(\mathsf{A}_2 \odot^T \mathsf{B}_2\right) \times_3 \left(\mathsf{A}_3 \odot^T \mathsf{B}_3\right)$$

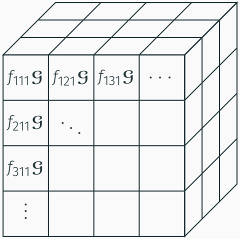
Operations with tensors in Tucker format

- ELEMENTWISE PRODUCT

$$\mathcal{X} = \mathcal{F} \times_1 \mathrm{A}_1 \times_2 \mathrm{A}_2 \times_3 \mathrm{A}_3, \quad \mathcal{F} \in \mathbb{R}^{P_1 \times P_2 \times P_3},$$
$$\mathcal{Y} = \mathcal{G} \times_1 \mathrm{B}_1 \times_2 \mathrm{B}_2 \times_3 \mathrm{B}_3, \quad \mathcal{G} \in \mathbb{R}^{Q_1 \times Q_2 \times Q_3},$$

$$\mathcal{X} * \mathcal{Y} = (\mathcal{F} \otimes \mathcal{G}) \times_1 \left(\mathrm{A}_1 \odot^T \mathrm{B}_1\right) \times_2 \left(\mathrm{A}_2 \odot^T \mathrm{B}_2\right) \times_3 \left(\mathrm{A}_3 \odot^T \mathrm{B}_3\right)$$

$$(\mathcal{F} \otimes \mathcal{G}) =$$



$$\in \mathbb{R}^{P_1 Q_1 \times P_2 Q_2 \times P_3 Q_3}$$

Direct sum using linear indexing

$$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ldots J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \oplus \mathcal{Y} \quad \ldots (I_1 + J_1) \times (I_2 + J_2) \times \cdots \times (I_N + J_N)$$

$$z_{k_1 k_2 \ldots k_N} = \begin{cases} x_{k_1 k_2 \ldots k_N}, & k_n = 1, 2, \ldots, I_n \\ y_{k_1 - I_1, k_2 - I2, \ldots, k_N - I_n}, & k_n = I_n + 1, \ldots, I_n + J_n \end{cases}$$

Direct sum using linear indexing

$$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ldots J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \oplus \mathcal{Y} \quad \ldots (I_1 + J_1) \times (I_2 + J_2) \times \cdots \times (I_N + J_N)$$

$$z_{k_1 k_2 \ldots k_N} = \begin{cases} x_{k_1 k_2 \ldots k_N}, & k_n = 1, 2, \ldots, I_n \\ y_{k_1 - I_1, k_2 - I2, \ldots, k_N - I_n}, & k_n = I_n + 1, \ldots, I_n + J_n \end{cases}$$

```
function directsum(X::Array{T1,N},Y::Array{T2,N}) where T1<:Number,T2<:Number,N
  I=size(X)
  J=size(Y)
  K=I.+J
  T1==T2 ? Z=zeros(T1,K) : Z=zeros(K)
  for i=1:prod(I)
      i_sub=ind2sub(X,i)
      Z[sub2ind(K,i_sub...)]=X[i]
  end
  for j=1:prod(J)
      j_sub=ind2sub(Y,j).+I
      Z[sub2ind(K,j_sub...)]=Y[j]
  end
  Z
end
```

Kronecker product using linear indexing

$$\mathcal{X} \ \ldots \ I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ \ldots \ J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y} \quad \ldots \ I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N$$

$$z_{k_1 k_2 \cdots k_N} = x_{i_1 i_2 \cdots i_N} y_{j_1 j_2 \cdots j_N}, \quad k_n = j_n + (i_n - 1) J_n$$

Kronecker product using linear indexing

$$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ldots J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y} \quad \ldots I_1J_1 \times I_2J_2 \times \cdots \times I_NJ_N$$

$$z_{k_1k_2\cdots k_N} = x_{i_1i_2\cdots i_N} y_{j_1j_2\cdots j_N}, \quad k_n = j_n + (i_n - 1)J_n$$

```julia
function tkron(X::Array{T1,N},Y::Array{T2,N}) where T1<:Number,T2<:Number,N
  I=size(X)
  J=size(Y)
  K=I.*J
  T1==T2 ? Z=zeros(T1,K) : Z=zeros(K)
  for i=1:prod(I)
    for j=1:prod(J)
      i_sub=ind2sub(I,i)
      j_sub=ind2sub(J,j)
      k=j_sub.+(i_sub.-1).*J
      Z[sub2ind(K,k...)]=X[i]*Y[j]
    end
  end
  Z
end
```

Advantages of using Julia

- `CartesianRange` $\rightarrow$ new type of iterator
- `CartesianIndex` $\rightarrow$ array indexing mechanism

Advantages of using Julia

- `CartesianRange` → new type of iterator
- `CartesianIndex` → array indexing mechanism

Example:

```
R=CartesianRange((4,3,2))
for I in R
   @show I
 end
```

```
I=CartesianIndex3((1,1,1))
I=CartesianIndex3((2,1,1))
I=CartesianIndex3((3,1,1))
I=CartesianIndex3((4,1,1))
I=CartesianIndex3((1,2,1))
I=CartesianIndex3((2,2,1))
I=CartesianIndex3((3,2,1))
⋮
```

```
I=first(R)
```

```
I=CartesianIndex{3}((1,1,1))
```

```
I=last(R)
```

```
I=CartesianIndex{3}((4,3,2))
```

Direct sum using CartesianRange

$$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ldots J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \oplus \mathcal{Y} \quad \ldots (I_1 + J_1) \times (I_2 + J_2) \times \cdots \times (I_N + J_N)$$

$$z_{k_1 k_2 \ldots k_N} = \begin{cases} x_{k_1 k_2 \ldots k_N}, & k_n = 1, 2, \ldots, I_n \\ y_{k_1 - I_1, k_2 - I2, \ldots, k_N - I_n}, & k_n = I_n + 1, \ldots, I_n + J_n \end{cases}$$

```
function directsum(X::Array{T1,N},Y::Array{T2,N}) where T1<:Number,T2<:Number,N
  I=size(X)
  J=size(Y)
  T1==T2 ? Z=zeros(T1,I.+J) : Z=zeros(I.+J)
  Rx=CartesianRange(I)
  for k in Rx
      Z[k]=X[k]
  end
  Il=last(Rx)
  Ry=CartesianRange(J)
  for k in Ry
      Z[k+Il]=Y[k]
  end
  Z
end
```

Kronecker product using CartesianRange

$$\mathcal{X} \ldots I_1 \times I_2 \times \cdots \times I_N, \quad \mathcal{Y} \ldots J_1 \times J_2 \times \cdots \times J_N$$

$$\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y} \quad \ldots I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N$$

$$z_{k_1 k_2 \cdots k_N} = x_{i_1 i_2 \cdots i_N} y_{j_1 j_2 \cdots j_N}, \quad k_n = j_n + (i_n - 1) J_n$$

```julia
function tkron(X::Array{T1,N},Y::Array{T2,N}) where T1<:Number,T2<:Number,N
  I=size(X)
  J=size(Y)
  T1==T2 ? Z=zeros(T1,I.*J) : Z=zeros(I.*J)
  Rx=CartesianRange(I)
  Ry=CartesianRange(J)
  Jl=last(Ry)
  for i in Rx
    for j in Ry
       Z[j+(i-1).*Jl]=X[i]*Y[j]
    end
  end
  Z
end
```

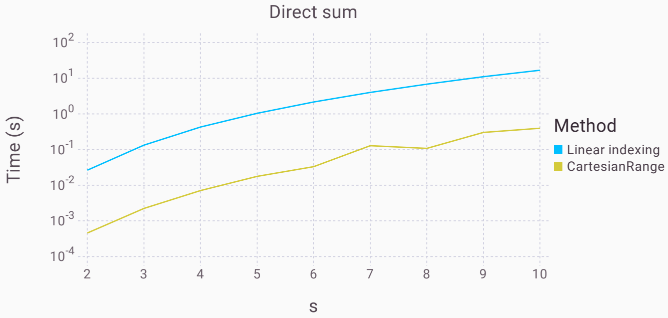
Comparison - Direct sum

$s = 2, 3, \ldots, 10$

$\mathfrak{X} \ldots s \times s \times s \times s$

$\mathfrak{Y} \ldots 5s \times 5s \times 5s \times 5s$

Execution times for getting $\mathfrak{Z} = \mathfrak{X} \oplus \mathfrak{Y}$ using linear indexing vs. CartesianRange:



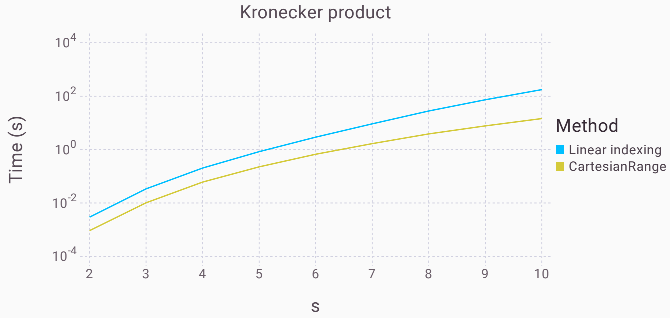
Comparison - Kronecker product

$s = 2, 3, \ldots, 10$

$\mathcal{X} \ldots s \times s \times s$

$\mathcal{Y} \ldots 5s \times 5s \times 5s$

Execution times for getting $\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y}$ using linear indexing vs. CartesianRange:



Conclusion

Julia provides powerful tools for creating multidimensional algorithms.

More about my work...

PACKAGE:

- TensorToolbox - official Julia package
- url: `https://github.com/lanaperisa/TensorToolbox.jl`

PAPER:

- D. Kressner and L. Periša - **Recompression of Hadamard Products of Tensors in Tucker Format**, SIAM Journal for Scientific computing, 2017

More about my work...

PACKAGE:

- TensorToolbox - official Julia package
- url: `https://github.com/lanaperisa/TensorToolbox.jl`

PAPER:

- D. Kressner and L. Periša - **Recompression of Hadamard Products of Tensors in Tucker Format**, SIAM Journal for Scientific computing, 2017

Thank you!