Table of Contents

Tensor Algebra Using MATLAB

```
classdef tensor
    methods(Static)
```

Vectorize

```
% This function computes the vec operator of a given matrix or tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: May 2022

function Y = vec(X)
    Y = reshape(X,[],1);
end
```

Hadamard Product

```
% This function computes the Hadarmard Product of two given matrices.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: January 2022

function C = mtx_prod_had(A,B)

    [ia,ja] = size(A);
    [ib,jb] = size(B);

    if (ia ~= ib) || (ja~=jb)
        disp('Invalid Matrices!')
        return;
    else
        C = A.*B;
        %C = zeros(ia,ja);
        %for i = 1:ia
            %for j = 1:ja
                %C(i,j) = A(i,j)*B(i,j);
            %end
        %end
    end
end
```

Kronecker Product

```
% This function computes the Kronecker Product of two given matrices.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: January 2022

function C = mtx_prod_kron(A,B)

    [ia,ja] = size(A);
    [ib,jb] = size(B);

    A = repelem(A,ib,jb);
    B = repmat(B,[ia ja]);
    C = A.*B;

end
```

Khatri-Rao Product

```
% This function computes the Khatri-Rao Product of two given matrices.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: January 2022

function C = mtx_prod_kr(A,B)
    [ia,ja] = size(A);
    [ib,jb] = size(B);

    if (ja~=jb)
        disp('Invalid Matrices!')
        return;
    else
        C = zeros(ia*ib,ja);
        for j = 1:ja
            C(:,j) = tensor.mtx_prod_kron(A(:,j),B(:,j));
        end
    end
end
```

Least-Squares Khatri-Rao Factorization (LSKRF)

```
% This function computes the LSKRF of a given matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: February 2022

function [Ahat,Bhat] = LSKRF(C,ia,ib)
    [~, jc] = size(C);

    Ahat = complex(zeros(ia,jc),0);
    Bhat = complex(zeros(ib,jc),0);

    for j = 1:jc
        Cp = C(:,j);
        Cp = reshape(Cp, [ib ia]);
        [U,S,V] = svd(Cp);
        Ahat(:,j) = sqrt(S(1,1)).*conj(V(:,1));
        Bhat(:,j) = sqrt(S(1,1)).*U(:,1);
    end
end
```

Least-Square Kronecker Product Factorization (LSKronF)

```
% This function computes the LSKronF of a given matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: February 2022

function [Ahat,Bhat] = LSKronF(C,ia,ja,ib,jb)
    [ic,jc] = size(C);

    I = (ic/ia) + zeros(1,ia);
    J = (jc/ja) + zeros(1,ja);
    blocks_of_C = mat2cell(C,I,J);

    k = 1;
    Chat = complex(zeros(ib*jb,ia*ja),0);
    for j = 1:ja
        for i = 1:ia
            vec_of_block = cell2mat(blocks_of_C(i,j));
            vec_of_block = vec_of_block(:);
            Chat(:,k) = vec_of_block;
            k = k + 1;
        end
    end

    [U,S,V] = svd(Chat);
    ahat = sqrt(S(1,1)).*conj(V(:,1));
    bhat = sqrt(S(1,1)).*U(:,1);
    Ahat = reshape(ahat,[ia ja]);
    Bhat = reshape(bhat, [ib jb]);
end
```

Kronecker Product Single Value Decomposition (KPSVD)

```
% This function computes the LSKRF of a given matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: March 2022

function [U,S,V,rkp] = KPSVD(X,ia,ja,ib,jb)
    [ix,jx] = size(X);

    I = (ix/ia) + zeros(1,ia);
    J = (jx/ja) + zeros(1,ja);
    blocks_of_X = mat2cell(X,I,J);

    k = 1;
    Xhat = complex(zeros(ib*jb,ia*ja),0);
    for j = 1:ja
        for i = 1:ia
            vec_of_block = cell2mat(blocks_of_X(i,j));
            vec_of_block = vec_of_block(:);
            Xhat(:,k) = vec_of_block;
            k = k + 1;
        end
    end
    [U,S,V] = svd(Xhat');
    rkp = sum(sum(S>0));
end
```

Unfolding

```
% This function computes the unfolding of a given tensor in its
 matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

function [A] = unfold(ten,mode)
    dim = size(ten);
    order = 1:numel(dim);
    order(mode) = [];
    order = [mode order];
    A = reshape(permute(ten,order), dim(mode), prod(dim)/dim(mode));
end
```

Folding

```
% This function computes the folding of a given matrix into its
 tensor.
```

```
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

% It's interesting to see how the dimmensions get swapped by the
 unfolding
% so the understanding of the code is clear.
function [ten] = fold(A,dim,mode)
    order = 1:numel(dim);
    order(mode) = [];
    order = [mode order];
    dim = dim(order);
    ten = reshape(A,dim);

    if mode == 1
        ten = permute(ten,order);
    else
        order = 1:numel(dim);
        for i = 2:mode
            order([i-1 i]) = order([i i-1]);
        end
        ten = permute(ten,order);
    end
end
```

N-mode Product

```
% This function computes n-mode product of a set of matrices and a
 tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

function [ten] = n_mod_prod(ten,matrices,modes)
    dim = size(ten);
    number = numel(matrices);
    if nargin < 3
       modes = 1:number;
    end

    for i = modes
        ten = cell2mat(matrices(i))*tensor.unfold(ten,i);
        [aux,~] = size(cell2mat(matrices(i)));
        dim(i) = aux;
        ten = tensor.fold(ten,[dim],i);
    end
end
```

Full High Order Single Value Decomposition (HOSVD)

```
% This function computes the Truncated HOSVD of a given tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

function [S,U] = HOSVD_full(ten)
    number = numel(size(ten));
    for i = 1:number
       [aux,~,~] = svd(tensor.unfold(ten,i));
       %[aux,~,~] = svd(tens2mat(ten,i));
       U{i} = aux;
    end
    % Core tensor uses the hermitian operator.
    Ut = cellfun(@(x) x', U,'UniformOutput',false);
    S = tensor.n_mod_prod(ten,Ut);
    % The normal factors should be transposed.
    U = cellfun(@(x) x, U,'UniformOutput',false);
end
```

Truncated High Order Single Value Decomposition (HOSVD)

```
% This function computes the Truncated HOSVD of a given tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

function [S,U] = HOSVD_truncated(ten,ranks)
    if nargin < 2
        number = numel(size(ten));
        for i = 1:number
           [aux,eig,~] = svd(tensor.unfold(ten,i));
           [aa,~] = size(eig(eig > eps));
           aux = aux(:,1:aa);
           U{i} = aux;
        end
        % Core tensor uses the hermitian operator.
        Ut = cellfun(@(x) x', U,'UniformOutput',false);
        S = tensor.n_mod_prod(ten,Ut);
        % The normal factors should be transposed.
        U = cellfun(@(x) x, U,'UniformOutput',false);
    else
        number = numel(size(ten));
        for i = 1:number
           [aux,~,~] = svd(tensor.unfold(ten,i));
           aux = aux(:,1:ranks(i));
           U{i} = aux;
        end
        % Core tensor uses the hermitian operator.
        Ut = cellfun(@(x) x',U,'UniformOutput',false);
        S = tensor.n_mod_prod(ten,Ut);
        % The normal factors should be transposed.
        U = cellfun(@(x) x,U,'UniformOutput',false);
    end
end
```

High Order Orthogonal Iteration (HOOI)

```
% This function computes the HOOI of a given tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: May 2022

function [S,U,k] = HOOI_full(ten)
    max_iter = 10;
    [~, U] = tensor.HOSVD_full(ten);
    number = numel(size(ten));
    for k = 1:max_iter
        for i = 1:number
            modes = 1:number;
            modes(i) = []; % It will skip this mode in the n_mod_prod.
            Un = tensor.n_mod_prod(ten,U,modes);
            [aux,~,~] = svd(tensor.unfold(Un,i));
            U{i} = aux;
        end
    end
    % The conjugate transpose
    Ut = cellfun(@(x) x', U ,'UniformOutput',false);
    S = tensor.n_mod_prod(ten,Ut);
end
```

Truncated High Order Orthogonal Iteration (HOOI)

```
% This function computes the Truncated HOOI of a given tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: May 2022

function [S,U] = HOOI_truncated(ten,ranks)
    if nargin < 2
        max_iter = 10;
        [~, U] = tensor.HOSVD_full(ten);
        number = numel(size(ten));
        for k = 1:max_iter
            for i = 1:number
                modes = 1:number;
                modes(i) = []; % It will skip this mode in the
 n_mod_prod.
                Un = tensor.n_mod_prod(ten,U,modes);
                [aux,eig,~] = svd(tensor.unfold(Un,i));
                [aa,~] = size(eig(eig > eps));
                U{i} = aux(:,1:aa);
            end
        end
        % The conjugate transpose
        Ut = cellfun(@(x) x', U ,'UniformOutput',false);
        S = tensor.n_mod_prod(ten,Ut);
    else
```

```
        max_iter = 10;
        [~, U] = tensor.HOSVD_full(ten);
        number = numel(size(ten));
        for k = 1:max_iter
            for i = 1:number
                modes = 1:number;
                modes(i) = []; % It will skip this mode in the
 n_mod_prod.
                Un = tensor.n_mod_prod(ten,U,modes);
                [aux,~,~] = svd(tensor.unfold(Un,i));
                U{i} = aux(:,1:ranks(i));
            end
        end
        % The conjugate transpose
        Ut = cellfun(@(x) x', U ,'UniformOutput',false);
        S = tensor.n_mod_prod(ten,Ut);
    end
end
```

Multidimensional Least-Squares Khatri-Rao Factorization (MLS-KRF)

```
% This function computes the MLS-KRF of a given matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: March 2022

% The dimensions should be inserted in the order that the products are
% performed.
function [A] = MLSKRF(X,N,dim)
    [~,R] = size(X);
    for r = 1:R
        xr = X(:,r);
        tenXr = reshape(xr,flip(dim));

        % Aplicar SVDs consecutivas em estrategia recursiva? Como
 lidar com
        % o nd  array nesse caso?
        [Sr,Ur] = tensor.HOSVD_full(tenXr);
        for n = 1:N
            Ar{r,n} = (Sr(1)^(1/N))*Ur{N - n + 1}(:,1);
        end
    end

    for n = 1:N
       aux = cell2mat(Ar(:,n));
       A{n} = reshape(aux,[dim(n) R]);
    end
end
```

Multidimensional Least-Squares Kronecker Factorization (MLS-KronF)

```
% This function computes the MLS-KronF of a given matrix.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: March 2022

% It is interesting to note that this process could easily be applied
 to a
% random number of matrices in a iterative form considering groups of
 3 objects.
function [Ahat] = MLSKronF(X,rows,columns)
    % 3rd structure
    %[ix,jx] = size(X);

    % 2nd structure
    I = rows(2)*rows(3) + zeros(1,rows(1));
    J = columns(2)*columns(3) + zeros(1,columns(1));
    blocks_of_X = mat2cell(X,I,J);

    % 1st structure
    Z = 1;
    for j = 1:columns(1)
        for i = 1:rows(1)
           aux = cell2mat(blocks_of_X(i,j));
           I = rows(3) + zeros(1,rows(2));
           J = columns(3) + zeros(1,columns(2));
           blocks_of_aux = mat2cell(aux,I,J);
           for jj = 1:columns(2)
                for ii = 1:rows(2)
                    vec_of_block = cell2mat(blocks_of_aux(ii,jj));
                    vec_of_block = vec_of_block(:);
                    mtx_1st(:,ii,jj) = vec_of_block;
                end
           end
            Xhat(:,Z) = reshape(mtx_1st,[],1);
            Z = Z + 1;
        end
    end

    tenXhat = reshape(Xhat,[rows(3)*columns(3), rows(2)*columns(2),
 rows(1)*columns(1)]);
    [S,U] = tensor.HOSVD_full(tenXhat);

    rows = flip(rows);
    columns = flip(columns);
    for u = 1:length(U)
        index = length(U) - u + 1;
        aux = (S(1)^(1/length(U)))*U{u}(:,1);
        Ahat{index} = reshape(aux,[rows(u) columns(u)]);
    end
end
```

Alternate Least-Square (ALS)

```
% This function computes the ALS of a given tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: April 2022

function [Ahat,Bhat,Chat,error] = ALS(X,R)
    I = zeros(R,R,R);
    for i = 1:R
        I(i,i,i) = 1;
    end
    [ia,ib,ic] = size(X);

    mode_1 = tensor.unfold(X,1);
    mode_2 = tensor.unfold(X,2);
    mode_3 = tensor.unfold(X,3);

    Ahat = randn(ia,R) + 1j*randn(ia,R);
    Bhat = randn(ib,R) + 1j*randn(ib,R);
    Chat = randn(ic,R) + 1j*randn(ic,R);

    aux = 1000;
    error = zeros(1,aux);
    error(1) = ((norm((mode_1 -
 Ahat*(tensor.mtx_prod_kr(Chat,Bhat).')),'fro'))^2)/
((norm(mode_1,'fro')^2));
    for i = 2:aux
        Bhat = mode_2*pinv((tensor.mtx_prod_kr(Chat,Ahat)).');
        Chat = mode_3*pinv((tensor.mtx_prod_kr(Bhat,Ahat)).');
        Ahat = mode_1*pinv((tensor.mtx_prod_kr(Chat,Bhat)).');
        error(i) = ((norm((mode_1 -
 Ahat*(tensor.mtx_prod_kr(Chat,Bhat).')),'fro'))^2)/
((norm(mode_1,'fro')^2));
        if abs(error(i) - error(i-1)) < eps
            error = error(1:i);
            break;
        else
            continue;
        end
    end
end
```

Tensor Kronecker Product

```
% This function computes the Tensor Kronecker Product of two given
 tensors.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: June 2022

function C = ten_prod_kron(A,B)

    aux1 = num2cell([size(A)]);
    aux2 = num2cell([size(B)]);

    A = repelem(A,aux2{:});
    B = repmat(B,aux1{:});
    C = A.*B;

end
```

Tensor Kronecker Product Single Value Decomposition (TKPSVD)

```
% This function computes the TKPSVD of a tensor.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: June 2022

function [Ahat,Bhat,Chat] = TKPSVD(tenX,tenSize,tenDim,N,R)
    % First reorder
    var = 1:length(tenSize);
    for n = 1:N
        tenXhatreorder{n} =
 cell2mat(tenSize(var(n:N:length(tenSize))));
    end
    tenXhatreorder = cell2mat(tenXhatreorder);
    tenXhat = reshape(tenX,tenXhatreorder);
    % Second reorder
    var = 1:length(tenSize);
    for n = 1:N
        tenXhatpermute{n} = var(n:N:length(tenSize));
    end
    tenXhatpermute = cell2mat(tenXhatpermute);
    tenXhat = permute(tenXhat,tenXhatpermute);
    % Third reorder
    tenXhat = reshape(tenXhat,tenDim{:});
    % ALS estimation
    [Ahat,Bhat,Chat,~] = tensor.ALS(tenXhat,R);
end
```

Tensor Contraction

```
% This function computes the tensor contraction between a tensor and a
 matrix
% or between two tensors.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: June 2022

function [tenZ] = contraction(tenX,n1,tenY,n2)
    % Obtaining the unfolds
    tenX_n = tensor.unfold(tenX,n1);
    tenY_n = tensor.unfold(tenY,n2);
    % Obtaining the unfold of the contracted tensor
    tenZ_n = (tenX_n.')*tenY_n;
    % Obtaining the fold of the contracted tensor
    dim_x = size(tenX);
    dim_x(n1) = [];
    dim_y = size(tenY);
    dim_y(n2) = [];
    dim_z = [dim_x dim_y];
    tenZ   = tensor.fold(tenZ_n,dim_z,1);
end
```

Tensor Train Single Value Decomposition (TT-SVD)

```
% This function computes the TT-SVD of a fourth order tensor but can
 be extended later.
% Author: Kenneth B. dos A. Benicio <kenneth@gtel.ufc.br>
% Created: June 2022

function [G] = TTSVD(tenX,Ranks)
    X_size = size(tenX);
    % Step 1
    X1 = tensor.unfold(tenX,1);
    [U1,S1,V1] = svd(X1,'econ');
    G1 = U1*sqrt(S1);
    G1 = G1(:,1:Ranks(1));
    G{1} = G1;
    % Step 2
    V1 = sqrt(S1)*V1';
    V1 = V1(1:Ranks(1),:);
    X2 = reshape(V1,[Ranks(1)*X_size(2) X_size(3)*X_size(4)]);
    [U2,S2,V2] = svd(X2,'econ');
    G2 = U2*sqrt(S2);
    G2 = G2(:,1:Ranks(2));
    G{2} = reshape(G2, [Ranks(1) X_size(2) Ranks(2)]);
    % Step 3
    V2 = sqrt(S2)*V2';
    V2 = V2(1:Ranks(2),:);
    X3 = reshape(V2,[Ranks(2)*X_size(3) X_size(4)]);
    [U3,S3,V3] = svd(X3,'econ');
    G3 = U3*sqrt(S3);
    G3 = G3(:,1:Ranks(3));
    G{3} = reshape(G3, [Ranks(2) X_size(3) Ranks(3)]);
    V3 = sqrt(S3)*V3';
    V3 = V3(1:Ranks(3),:);
    G{4} = V3;
end

    end
end
```

```
ans =

  tensor with no properties.
```

Published with MATLAB® R2021a