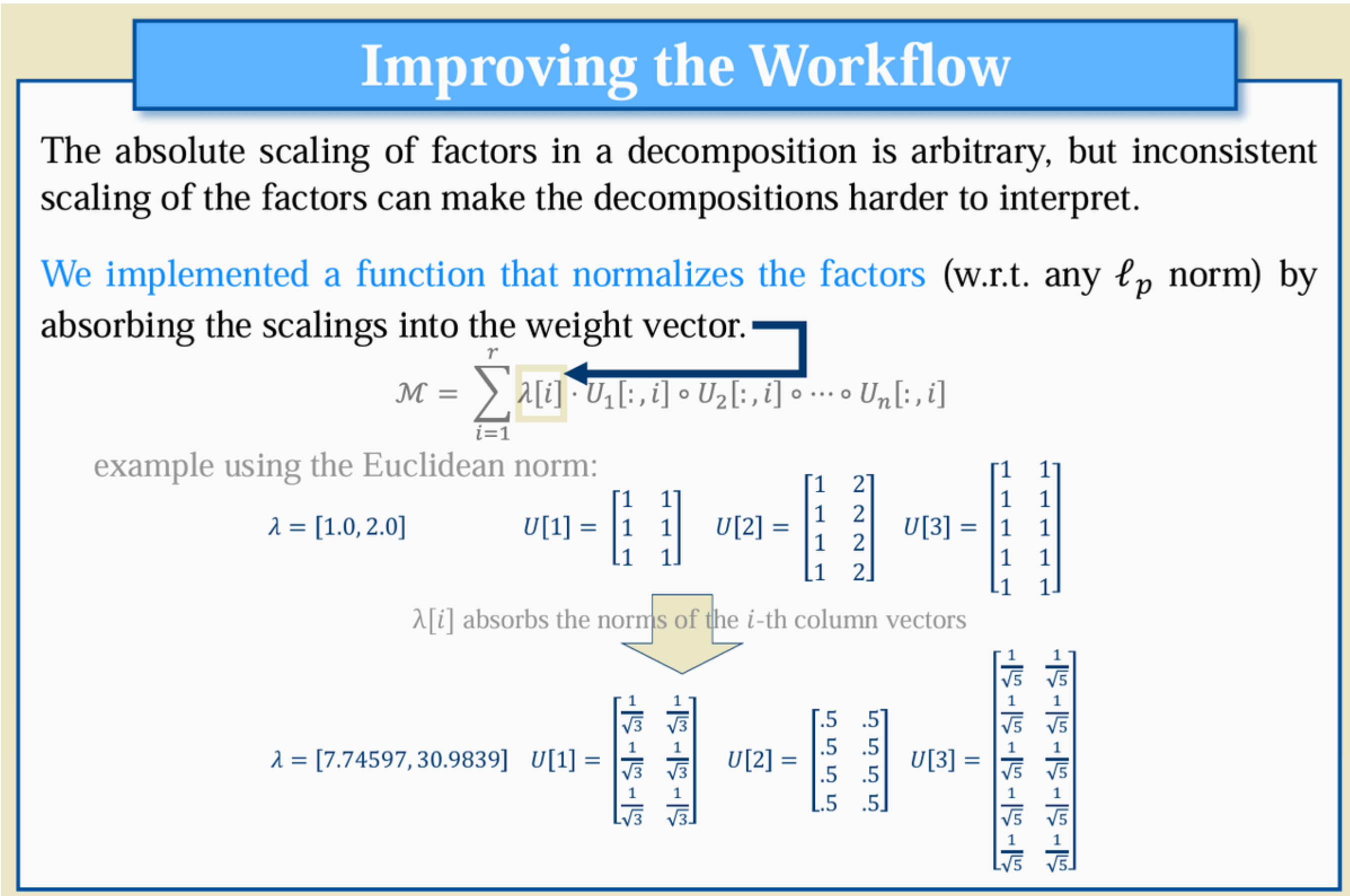
Normalization and Weight Absorption Function




```julia
using LinearAlgebra,GCPDecompositions, TensorCore
```

The Function

```
cpd_normalize (generic function with 2 methods)
```

```julia
function cpd_normalize(M::CPD, p::Real = 2)
    weights = M.λ[:]

    for matrix in 1:ndims(M)
        scaling = [norm(M.U[matrix][:, col], p) for col in 1:ncomponents(M)]

        weights .*= scaling

        M.U[matrix] .= M.U[matrix] ./ scaling'
    end

    M.λ[:] = weights

    return M
end
```

Usage and Examples

#1 - L1 norm (Manhattan Geometry)

this norm purely adds up all values in the column and uses the sum as the normalization factor

```
X = 3×4×5 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 1 component
    λ weights:
    1-element Vector{Float64}: …
    U[1] factor matrix:
    3×1 Matrix{Float64}: …
    U[2] factor matrix:
    4×1 Matrix{Float64}: …
    U[3] factor matrix:
    5×1 Matrix{Float64}: …
```

```julia
X = CPD([1.0], (ones(3,1), ones(4,1), ones(5,1)))
```

```
[1.0]
```

```julia
X.λ
```

```
(
    1:  3×1 Matrix{Float64}:
         1.0
         1.0
         1.0
    2:  4×1 Matrix{Float64}:
         1.0
         1.0
         1.0
         1.0
    3:  5×1 Matrix{Float64}:
         1.0
         1.0
         1.0
         1.0
         1.0
)
```

```
X.U[:]
```

```
cpd_normalize(X,1);
```

```
[60.0]
```

```
X.λ
```

```
(
    1:  3×1 Matrix{Float64}:
         0.333333
         0.333333
         0.333333
    2:  4×1 Matrix{Float64}:
         0.25
         0.25
         0.25
         0.25
    3:  5×1 Matrix{Float64}:
         0.2
         0.2
         0.2
         0.2
         0.2
)
```

```
X.U[:]
```

Divided the first matrix by 3, second by 4, and third by 5.

#2 - L2 norm (Euclidean)

the default norm (sum the squares of each element and take the square root)

```
Y = 3×4×5 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 2 components
    λ weights:
    2-element Vector{Float64}: …
    U[1] factor matrix:
    3×2 Matrix{Float64}: …
    U[2] factor matrix:
    4×2 Matrix{Float64}: …
    U[3] factor matrix:
    5×2 Matrix{Float64}: …
```

```
Y = CPD([1.0,2.0], (ones(3,2), ones(4,2), ones(5,2)))
```

```
[1.0, 2.0]
```

```
Y.λ
```

```
(
   1:  3×2 Matrix{Float64}:
        1.0  1.0
        1.0  1.0
        1.0  1.0
   2:  4×2 Matrix{Float64}:
        1.0  1.0
        1.0  1.0
        1.0  1.0
        1.0  1.0
   3:  5×2 Matrix{Float64}:
        1.0  1.0
        1.0  1.0
        1.0  1.0
        1.0  1.0
        1.0  1.0
)
```

```
Y.U[:]
```

```
3×4×5 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 2 components
λ weights:
2-element Vector{Float64}: …
U[1] factor matrix:
3×2 Matrix{Float64}: …
U[2] factor matrix:
4×2 Matrix{Float64}: …
U[3] factor matrix:
5×2 Matrix{Float64}: …
```

```
cpd_normalize(Y) # Does not need specification, function defaults to L2
```

```
[7.74597, 15.4919]
```

```
Y.λ # lambda weight absorption
```

```
(
  1: 3×2 Matrix{Float64}:
      0.57735  0.57735
      0.57735  0.57735
      0.57735  0.57735
  2: 4×2 Matrix{Float64}:
      0.5  0.5
      0.5  0.5
      0.5  0.5
      0.5  0.5
  3: 5×2 Matrix{Float64}:
      0.447214  0.447214
      0.447214  0.447214
      0.447214  0.447214
      0.447214  0.447214
      0.447214  0.447214
)
```

```
Y.U[:]
```

```
1.0
```

```
norm(Y.U[2][:,2])
```

#3 - Inf norm

this norm takes the biggest value in that column and uses it as the normalization factor

```
Z = 3×3×3 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 3 components
    λ weights:
    3-element Vector{Float64}: …
    U[1] factor matrix:
    3×3 Matrix{Float64}: …
    U[2] factor matrix:
    3×3 Matrix{Float64}: …
    U[3] factor matrix:
    3×3 Matrix{Float64}: …
```

```
Z =CPD([1.0,1.0,2.0],(float([6  4  2;
    12 8  4;
    18 12 6]),float([1  2  3;
    12 8  3;
    18 12 3]),float([1  1  1;
    1 1  1;
    1 1 1])))
```

```
[1.0, 1.0, 2.0]
```

```
Z.λ
```

```
(
  1: 3×3 Matrix{Float64}:
      6.0   4.0  2.0
     12.0   8.0  4.0
     18.0  12.0  6.0
  2: 3×3 Matrix{Float64}:
      1.0   2.0  3.0
     12.0   8.0  3.0
     18.0  12.0  3.0
  3: 3×3 Matrix{Float64}:
     1.0  1.0  1.0
     1.0  1.0  1.0
     1.0  1.0  1.0
)
```

```
Z.U[:]
```

```
cpd_normalize(Z,Inf);
```

```
[324.0, 144.0, 36.0]
```

```
Z.λ
```

```
(
  1: 3×3 Matrix{Float64}:
     0.333333  0.333333  0.333333
     0.666667  0.666667  0.666667
     1.0       1.0       1.0
  2: 3×3 Matrix{Float64}:
     0.0555556  0.166667  1.0
     0.666667   0.666667  1.0
     1.0        1.0       1.0
  3: 3×3 Matrix{Float64}:
     1.0  1.0  1.0
     1.0  1.0  1.0
     1.0  1.0  1.0
)
```

```
Z.U[:]
```

Functionality

This normalization process provides the user with greater analytical power by offering broader applicability, scale consistency, and numerical stability when dealing with multitudes of data.