Institut für Theoretische
Physik
Universität zu Köln

Prof. Dr. Simon Trebst
Peter Bröcker

Computerphysik
Vorlesung — Programmiertechniken 7

Sommersemester 2019

Website: http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml (http://www.thp.uni-koeln.de/trebst/Lectures/2019-CompPhys.shtml)

Lineare Algebra

0. Bibliothek

In [8]:
```
using LinearAlgebra
```

1. Vektoren und Matrizen

Vektoren

In [1]:
```
v1 = [1, 2, 3]
```

Out[1]:
```
3-element Array{Int64,1}:
 1
 2
 3
```

In [2]:
```
v2 = [4, 5, 6]
```

Out[2]:
```
3-element Array{Int64,1}:
 4
 5
 6
```

Skalarprodukt

In [3]:
```
v1*v2
```

```
MethodError: no method matching *(::Array{Int64,1}, ::Array{Int64,1}
)
Closest candidates are:
  *(::Any, ::Any, !Matched::Any, !Matched::Any...) at operators.jl:5
02
  *(!Matched::LinearAlgebra.Adjoint{#s549,#s548} where #s548<:Union{
DenseArray{T<:Union{Complex{Float32}, Complex{Float64}, Float32, Flo
at64},2}, ReinterpretArray{T<:Union{Complex{Float32}, Complex{Float6
4}, Float32, Float64},2,S,A} where S where A<:Union{SubArray{T,N,A,I
,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where
A<:DenseArray where N where T, DenseArray}, ReshapedArray{T<:Union{C
omplex{Float32}, Complex{Float64}, Float32, Float64},2,A,MI} where M
I<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} where N} where
A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubArray{T
,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N}
where A<:DenseArray where N where T, DenseArray} where N where T, Su
bArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N}
where N} where A<:DenseArray where N where T, DenseArray}, SubArray{
T<:Union{Complex{Float32}, Complex{Float64}, Float32, Float64},2,A,I
,L} where L where I<:Tuple{Vararg{Union{Int64, AbstractRange{Int64},
AbstractCartesianIndex},N} where N} where A<:Union{ReinterpretArray{
T,N,S,A} where S where A<:Union{SubArray{T,N,A,I,true} where I<:Tupl
e{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray where
N where T, DenseArray} where N where T, ReshapedArray{T,N,A,MI} wher
e MI<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} where N} wh
ere A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubArr
ay{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} wher
e N} where A<:DenseArray where N where T, DenseArray} where N where
T, SubArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{An
y,N} where N} where A<:DenseArray where N where T, DenseArray} where
N where T, DenseArray}} where #s549, ::Union{DenseArray{S,1}, Reinte
rpretArray{S,1,S,A} where S where A<:Union{SubArray{T,N,A,I,true} wh
ere I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:Dense
Array where N where T, DenseArray}, ReshapedArray{S,1,A,MI} where MI
<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} where N} where
A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubArray{T
,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N}
where A<:DenseArray where N where T, DenseArray} where N where T, Su
bArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N}
where N} where A<:DenseArray where N where T, DenseArray}, SubArray{
S,1,A,I,L} where L where I<:Tuple{Vararg{Union{Int64, AbstractRange{
Int64}, AbstractCartesianIndex},N} where N} where A<:Union{Reinterpr
etArray{T,N,S,A} where S where A<:Union{SubArray{T,N,A,I,true} where
I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArra
y where N where T, DenseArray} where N where T, ReshapedArray{T,N,A,
MI} where MI<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} whe
re N} where A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Unio
n{SubArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any
,N} where N} where A<:DenseArray where N where T, DenseArray} where
N where T, SubArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,V
ararg{Any,N} where N} where A<:DenseArray where N where T, DenseArra
y} where N where T, DenseArray}}) where {T<:Union{Complex{Float32},
```

```
Complex{Float64}, Float32, Float64}, S} at /Users/osx/buildbot/slave
/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/m
atmul.jl:97
  *(!Matched::LinearAlgebra.Adjoint{#s549,#s548} where #s548<:Linear
Algebra.AbstractTriangular where #s549, ::AbstractArray{T,1} where T
) at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/s
tdlib/v1.0/LinearAlgebra/src/triangular.jl:1805
  ...

Stacktrace:
 [1] top-level scope at In[3]:1
```

In [4]:
```
transpose(v1)
```

Out[4]:
```
1×3 LinearAlgebra.Transpose{Int64,Array{Int64,1}}:
 1  2  3
```

In [5]:
```
transpose(v1)*v2
```

Out[5]:
```
32
```

In [9]:
```
dot(v1,v2)
```

Out[9]:
```
32
```

In [10]:
```
⋅(v1,v2)
```

Out[10]:
```
32
```

Vektorprodukt

In [11]:
```
cross(v1,v2)
```

Out[11]:
```
3-element Array{Int64,1}:
 -3
  6
 -3
```

In [12]:
```
×(v1,v2)
```

Out[12]:
```
3-element Array{Int64,1}:
 -3
  6
 -3
```

Matrizen

In [14]:
```
M = rand(3,3)
```

Out[14]:
```
3×3 Array{Float64,2}:
 0.511945  0.12192   0.622005
 0.845555  0.185127  0.983774
 0.365761  0.914772  0.407481
```

In [15]:
```
N = rand(3,3)
```

Out[15]:
```
3×3 Array{Float64,2}:
 0.788633  0.192182   0.809607
 0.670056  0.272727   0.29281
 0.913363  0.0219473  0.407991
```

Matrix-Multiplikation

In [16]:
```
M*N
```

Out[16]:
```
3×3 Array{Float64,2}:
 1.05355  0.145289  0.703946
 1.68942  0.234581  1.14015
 1.27358  0.328719  0.730226
```

In [17]:
```
*(M,N)
```

Out[17]:
```
3×3 Array{Float64,2}:
 1.05355  0.145289  0.703946
 1.68942  0.234581  1.14015
 1.27358  0.328719  0.730226
```

2. Lineare Gleichungssyteme

Betrachten wir ein lineares Gleichungssystem mit drei Unbekannten, etwa

$$\begin{aligned} x - 2y + 2z &= 5 \\ x - y + 2z &= 7 \\ -x + y + z &= 5 \end{aligned}$$

In [18]:
```
A = [1 -2 2; 1 -1 2; -1 1 1]
```

Out[18]:
```
3×3 Array{Int64,2}:
  1  -2  2
  1  -1  2
 -1   1  1
```

In [19]:
```
b = [5, 7, 5]
```

Out[19]:
```
3-element Array{Int64,1}:
 5
 7
 5
```

Damit haben wir ein Gleichungssystem der Form

$$A \cdot \vec{x} = \vec{b},$$

welches wir nach dem unbekannten Vektor $\vec{x}$ auflösen wollen.

In [20]:
```
x = A\b
```

Out[20]:
```
3-element Array{Float64,1}:
 1.0
 2.0
 4.0
```

In [21]:
```
A * x
```

Out[21]:
```
3-element Array{Float64,1}:
 5.0
 7.0
 5.0
```

3. Determinanten und Inverse

In [22]:
```
det(A)
```

Out[22]:
```
3.0
```

In [23]:
```
inv(A)
```

Out[23]:
```
3×3 Array{Float64,2}:
 -1.0   1.33333   -0.666667
 -1.0   1.0       -0.0
  0.0   0.333333   0.333333
```

In [25]:
```
tr(A)
```

Out[25]:
```
1
```

In [26]:
```
rank(A)
```

Out[26]:
```
3
```

Betrachte ein **unterdeterminiertes** Gleichungssystem

In [27]:
```
A1 = [1 -2 2; 1 -2 2; -1 1 1]
```

Out[27]:
```
3×3 Array{Int64,2}:
  1  -2  2
  1  -2  2
 -1   1  1
```

In [28]:
```
det(A1)
```

Out[28]:
```
0.0
```

In [29]:
```
rank(A1)
```

Out[29]:
```
2
```

In [30]:
```
inv(A1)
```

```
SingularException(3)

Stacktrace:
 [1] checknonsingular at /Users/osx/buildbot/slave/package_osx64/bui
ld/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/factorization.jl:12
[inlined]
 [2] #lu!#97(::Bool, ::Function, ::Array{Float64,2}, ::Val{true}) at
/Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib
/v1.0/LinearAlgebra/src/lu.jl:41
 [3] #lu! at ./none:0 [inlined]
 [4] #lu#101 at /Users/osx/buildbot/slave/package_osx64/build/usr/sh
are/julia/stdlib/v1.0/LinearAlgebra/src/lu.jl:142 [inlined]
 [5] lu at /Users/osx/buildbot/slave/package_osx64/build/usr/share/j
ulia/stdlib/v1.0/LinearAlgebra/src/lu.jl:142 [inlined] (repeats 2 ti
mes)
 [6] inv(::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx6
4/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/dense.jl:714
 [7] top-level scope at In[30]:1
```

In [31]:
```
A1\b
```

```
SingularException(3)

Stacktrace:
 [1] checknonsingular at /Users/osx/buildbot/slave/package_osx64/bui
ld/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/factorization.jl:12
[inlined]
 [2] #lu!#97(::Bool, ::Function, ::Array{Float64,2}, ::Val{true}) at
/Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib
/v1.0/LinearAlgebra/src/lu.jl:41
 [3] #lu! at ./none:0 [inlined]
 [4] #lu#103(::Bool, ::Function, ::Array{Int64,2}) at /Users/osx/bui
ldbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAl
gebra/src/lu.jl:251
 [5] lu at /Users/osx/buildbot/slave/package_osx64/build/usr/share/j
ulia/stdlib/v1.0/LinearAlgebra/src/lu.jl:242 [inlined]
 [6] \(::Array{Int64,2}, ::Array{Int64,1}) at /Users/osx/buildbot/sl
ave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/sr
c/generic.jl:870
 [7] top-level scope at In[31]:1
```

Betrachte ein **unterdeterminiertes** Gleichungssystem mit unendlich vielen Lösungen

In [32]:
```
A2 = [1 -2 2; 1 -1 2]
```

Out[32]:
```
2×3 Array{Int64,2}:
 1  -2  2
 1  -1  2
```

In [33]:
```
det(A2)
```

```
DimensionMismatch("matrix is not square: dimensions are (2, 3)")

Stacktrace:
 [1] checksquare at /Users/osx/buildbot/slave/package_osx64/build/us
r/share/julia/stdlib/v1.0/LinearAlgebra/src/LinearAlgebra.jl:216 [in
lined]
 [2] det(::LU{Float64,Array{Float64,2}}) at /Users/osx/buildbot/slav
e/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/
lu.jl:378
 [3] det(::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx6
4/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/generic.jl:124
2
 [4] top-level scope at In[33]:1
```

In [34]:
```
inv(A2)
```

```
DimensionMismatch("matrix is not square: dimensions are (2, 3)")

Stacktrace:
 [1] inv(::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/LinearAlgebra.jl:216
 [2] top-level scope at In[34]:1
```

In [35]:
```
b2 = [7, 5]
```

Out[35]:
```
2-element Array{Int64,1}:
 7
 5
```

In [36]:
```
A2\b2
```

Out[36]:
```
3-element Array{Float64,1}:
  0.5999999999999999
 -2.0
  1.2
```

Betrachte ein **überdeterminiertes** Gleichungssystem

In [37]:
```
A3 = [ -2 2;  -1 2;  1 1]
```

Out[37]:
```
3×2 Array{Int64,2}:
 -2  2
 -1  2
  1  1
```

In [38]:
```
A3\b
```

Out[38]:
```
2-element Array{Float64,1}:
 1.275862068965517
 3.931034482758621
```

In [ ]:
```

```

4. Eigenwerte und -vektoren

In [39]:
```
A = [1 -2 2; 1 -1 2; -1 1 1]
```

Out[39]:
```
3×3 Array{Int64,2}:
  1  -2  2
  1  -1  2
 -1   1  1
```

In [40]:
```
λ = eigvals(A)
```

Out[40]:
```
3-element Array{Complex{Float64},1}:
 -0.2873715369435107 + 1.3499963980036567im
 -0.2873715369435107 - 1.3499963980036567im
  1.5747430738870216 + 0.0im
```

In [41]:
```
v = eigvecs(A)
```

Out[41]:
```
3×3 Array{Complex{Float64},2}:
   0.783249+0.0im          0.783249-0.0im        0.237883+0.0im
   0.493483-0.303862im     0.493483+0.303862im   0.651777+0.0im
 -0.0106833+0.22483im    -0.0106833-0.22483im    0.720138+0.0im
```

In [42]:
```
v1 = v[:,1]
```

Out[42]:
```
3-element Array{Complex{Float64},1}:
    0.7832493829899281 + 0.0im
    0.4934831899521167 - 0.303862020899599im
 -0.010683291042783476 + 0.22482990198789593im
```

In [43]:
```
v2 = v[:,2]
```

Out[43]:
```
3-element Array{Complex{Float64},1}:
    0.7832493829899281 - 0.0im
    0.4934831899521167 + 0.303862020899599im
 -0.010683291042783476 - 0.22482990198789593im
```

In [44]:
```
v3 = v[:,3]
```

Out[44]:
```
3-element Array{Complex{Float64},1}:
 0.23788283104387067 + 0.0im
  0.6517770419563313 + 0.0im
  0.7201377967258821 + 0.0im
```

sanity check

In [45]:
```
A*v1
```

Out[45]:
```
3-element Array{Complex{Float64},1}:
  -0.2250835789998723 + 1.0573838457749898im
   0.2683996109522444 + 0.7535218248753909im
 -0.30044948408059485 - 0.07903211891170309im
```

In [46]:
```
λ[1]*v1
```

Out[46]:
```
3-element Array{Complex{Float64},1}:
 -0.22508357899987208 + 1.0573838457749896im
   0.2683996109522443 + 0.7535218248753909im
 -0.30044948408059513 - 0.07903211891170295im
```

In [47]:
```
A*v1 - λ[1]*v1
```

Out[47]:
```
3-element Array{Complex{Float64},1}:
 -2.220446049250313e-16 + 2.220446049250313e-16im
 1.1102230246251565e-16 + 0.0im
 2.7755575615628914e-16 - 1.3877787807814457e-16im
```

Spezielle Matrizen

In [48]:
```
A
```

Out[48]:
```
3×3 Array{Int64,2}:
  1  -2  2
  1  -1  2
 -1   1  1
```

In [49]:
```
diag(A)
```

Out[49]:
```
3-element Array{Int64,1}:
  1
 -1
  1
```

In [50]:
```
Diagonal(diag(A))
```

Out[50]:
```
3×3 Diagonal{Int64,Array{Int64,1}}:
 1   ·   ·
 ·  -1   ·
 ·   ·   1
```

In [ ]:
```
```

5. Matrix-Zerlegungen

Singulärwert-Zerlegung / singular value decomposition (SVD)

$$A = USV^\dagger$$

In [51]:
```
U, S, V = svd(A);
```

In [52]:
```
U
```

Out[52]:
```
3×3 Array{Float64,2}:
 -0.777773   -0.0889336  -0.622222
 -0.625027    0.214034    0.750687
  0.0664157   0.972769   -0.222056
```

In [53]:
```
S
```

Out[53]:
```
3-element Array{Float64,1}:
 3.8354496831938465
 1.7582443340259464
 0.4448624389110475
```

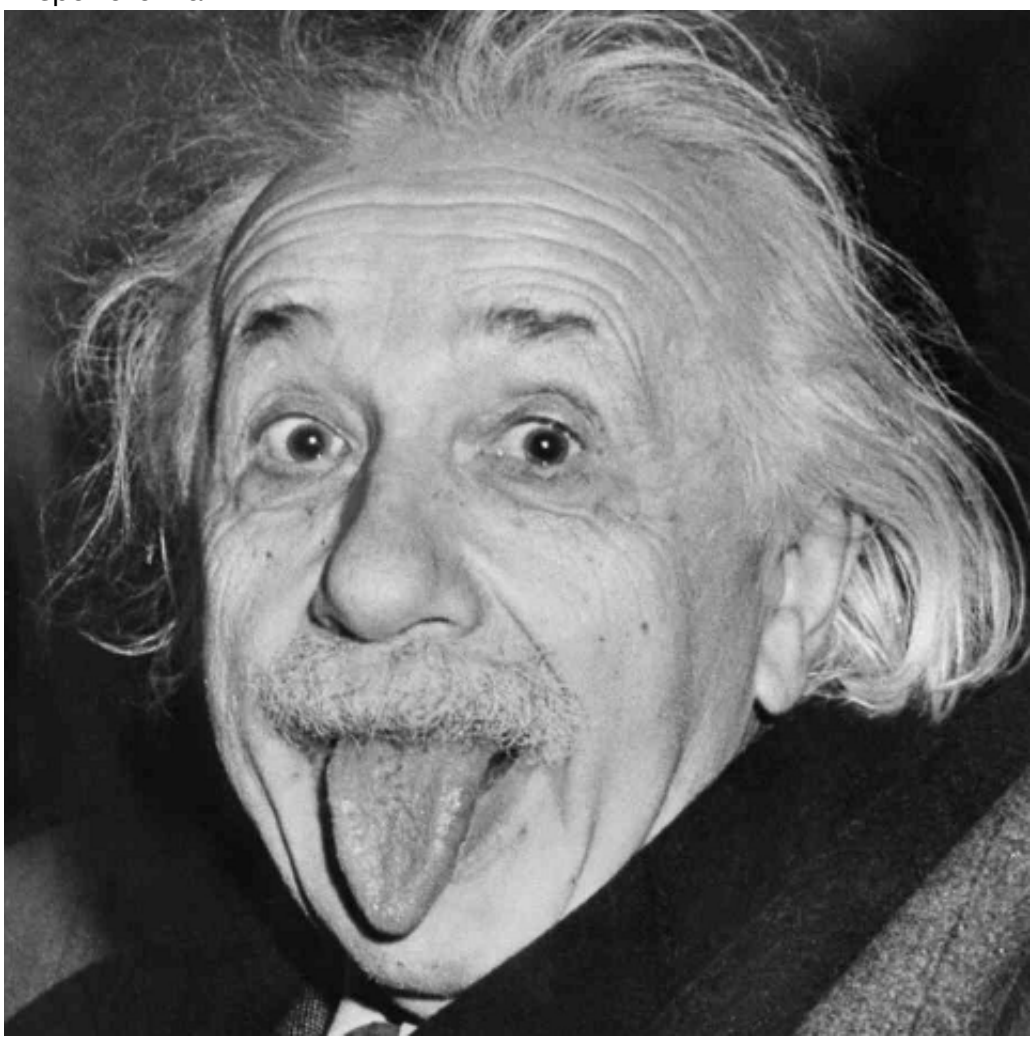
In [54]:
```
V
```

Out[54]:
```
3×3 Adjoint{Float64,Array{Float64,2}}:
 -0.383062  -0.482111  0.787929
  0.585847   0.532692  0.610756
 -0.714175   0.695564  0.0783893
```

In [55]:
```
svdvals(A)
```

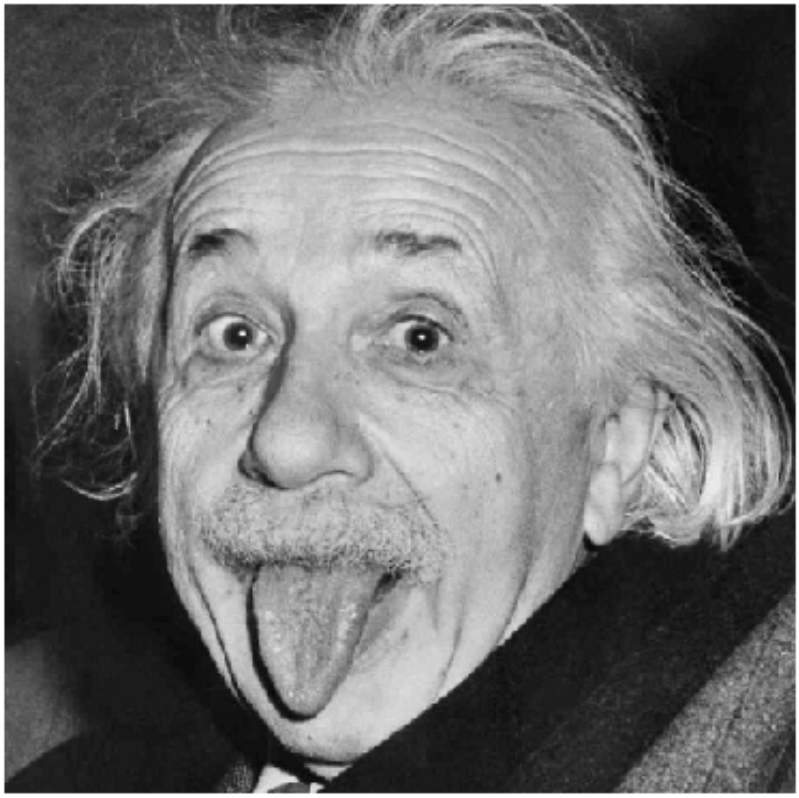
Out[55]:
```
3-element Array{Float64,1}:
 3.8354496831938456
 1.7582443340259462
 0.44486243891104754
```

Eine ganz spezielle Matrix



In [56]:
```
using PyPlot

Einstein = imread("http://www.thp.uni-koeln.de/trebst/Lectures/CompPhy
imshow(Einstein, cmap="gray")
axis("off")
```



Out[56]: (-0.5, 479.5, 479.5, -0.5)

In [57]:
```
Einstein
```

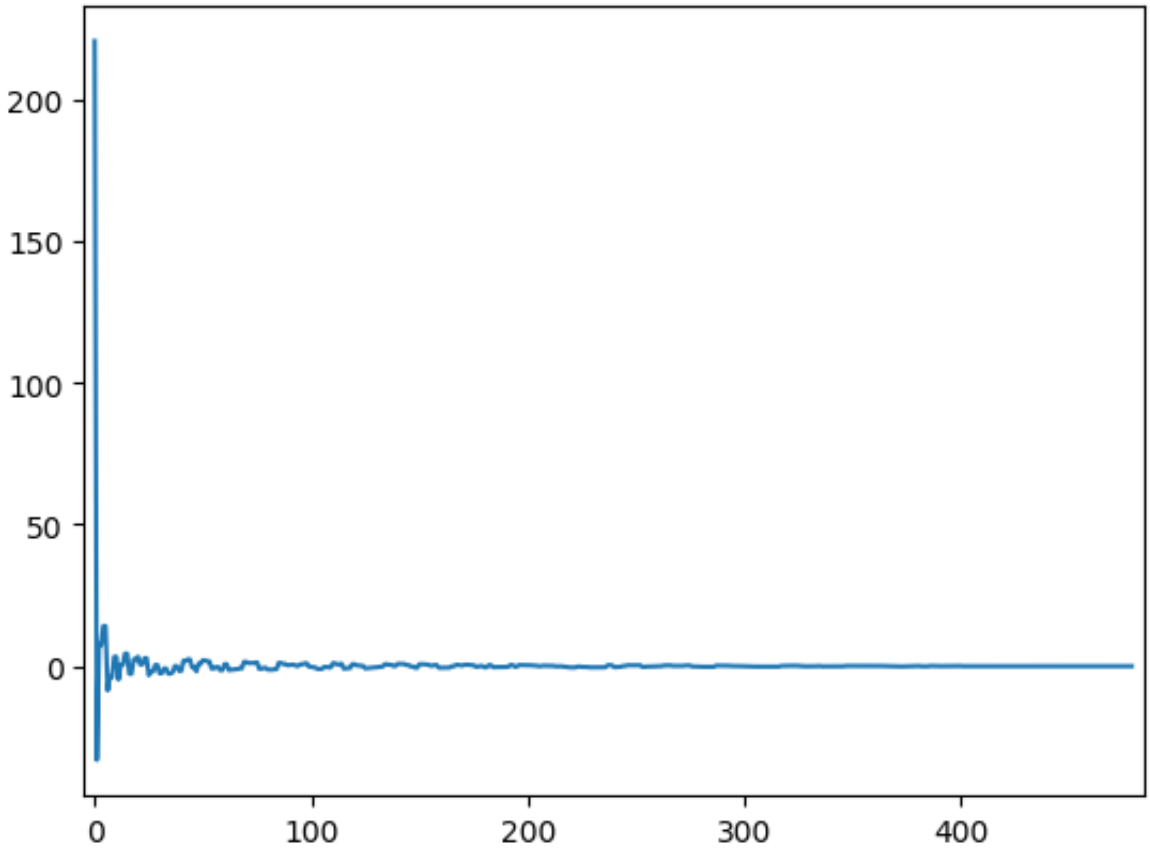
Out[57]:
```
480×480 Array{Float32,2}:
 0.235294  0.239216  0.25098   0.243137  …  0.0862745  0.0823529  0.
0901961
 0.239216  0.235294  0.247059  0.247059     0.0980392  0.0980392  0.
0901961
 0.239216  0.235294  0.258824  0.25098      0.0901961  0.0901961  0.
0901961
 0.239216  0.235294  0.270588  0.254902     0.0823529  0.0823529  0.
0901961
 0.235294  0.231373  0.278431  0.258824     0.0745098  0.0784314  0.
0901961
 0.254902  0.286275  0.270588  0.247059  …  0.0784314  0.0745098  0.
0745098
 0.258824  0.27451   0.290196  0.25098      0.0784314  0.0784314  0.
0784314
 0.262745  0.266667  0.27451   0.247059     0.0784314  0.0784314  0.
0784314
```

```
 0.266667  0.262745  0.235294  0.231373     0.0784314  0.0745098  0.
0745098
 0.270588  0.231373  0.215686  0.247059     0.0784314  0.0784314  0.
0901961
 0.258824  0.235294  0.247059  0.25098   …  0.0784314  0.0784314  0.
0901961
 0.239216  0.235294  0.258824  0.25098      0.0784314  0.0784314  0.
0901961
 0.227451  0.243137  0.231373  0.235294     0.0784314  0.0784314  0.
0901961
 ⋮                                       ⋱
 0.321569  0.321569  0.34902   0.364706     0.282353   0.25098    0.
180392
 0.352941  0.356863  0.360784  0.34902      0.301961   0.231373   0.
243137
 0.337255  0.321569  0.309804  0.266667  …  0.32549    0.290196   0.
262745
 0.317647  0.27451   0.235294  0.184314     0.298039   0.298039   0.
262745
 0.309804  0.247059  0.180392  0.133333     0.270588   0.282353   0.
239216
 0.215686  0.2       0.109804  0.113725     0.266667   0.239216   0.
278431
 0.156863  0.137255  0.113725  0.117647     0.254902   0.247059   0.
270588
 0.145098  0.137255  0.113725  0.117647  …  0.247059   0.25098    0.
294118
 0.12549   0.12549   0.113725  0.117647     0.231373   0.262745   0.
337255
 0.12549   0.117647  0.117647  0.117647     0.247059   0.282353   0.
32549
 0.12549   0.117647  0.117647  0.117647     0.27451    0.309804   0.
321569
 0.12549   0.117647  0.117647  0.117647     0.254902   0.286275   0.
309804
```

In [58]:
```
λ_einstein = eigvals(Einstein)
```

Out[58]:
```
480-element Array{Complex{Float32},1}:
     220.72752f0 + 0.0f0im
     -33.09746f0 + 0.0f0im
     7.4950356f0 + 13.796992f0im
     7.4950356f0 - 13.796992f0im
     14.175516f0 + 2.2396793f0im
     14.175516f0 - 2.2396793f0im
     -8.578165f0 + 0.0f0im
     -3.884768f0 + 4.4624815f0im
     -3.884768f0 - 4.4624815f0im
     3.3191054f0 + 4.7837715f0im
     3.3191054f0 - 4.7837715f0im
    -4.7141123f0 + 0.0f0im
     0.42903125f0 + 4.54011f0im
                  ⋮
   0.011390839f0 + 0.004110194f0im
   0.011390839f0 - 0.004110194f0im
  -0.005495188f0 + 0.014608204f0im
  -0.005495188f0 - 0.014608204f0im
  -0.013313696f0 + 0.010065446f0im
  -0.013313696f0 - 0.010065446f0im
  -0.019703597f0 + 0.0f0im
 -0.0034724171f0 + 0.010515782f0im
 -0.0034724171f0 - 0.010515782f0im
  -0.004805194f0 + 0.0054243365f0im
  -0.004805194f0 - 0.0054243365f0im
 -0.0033080133f0 + 0.0f0im
```

In [59]:
```
plot(λ_einstein)
xlim(-5, 485)
```

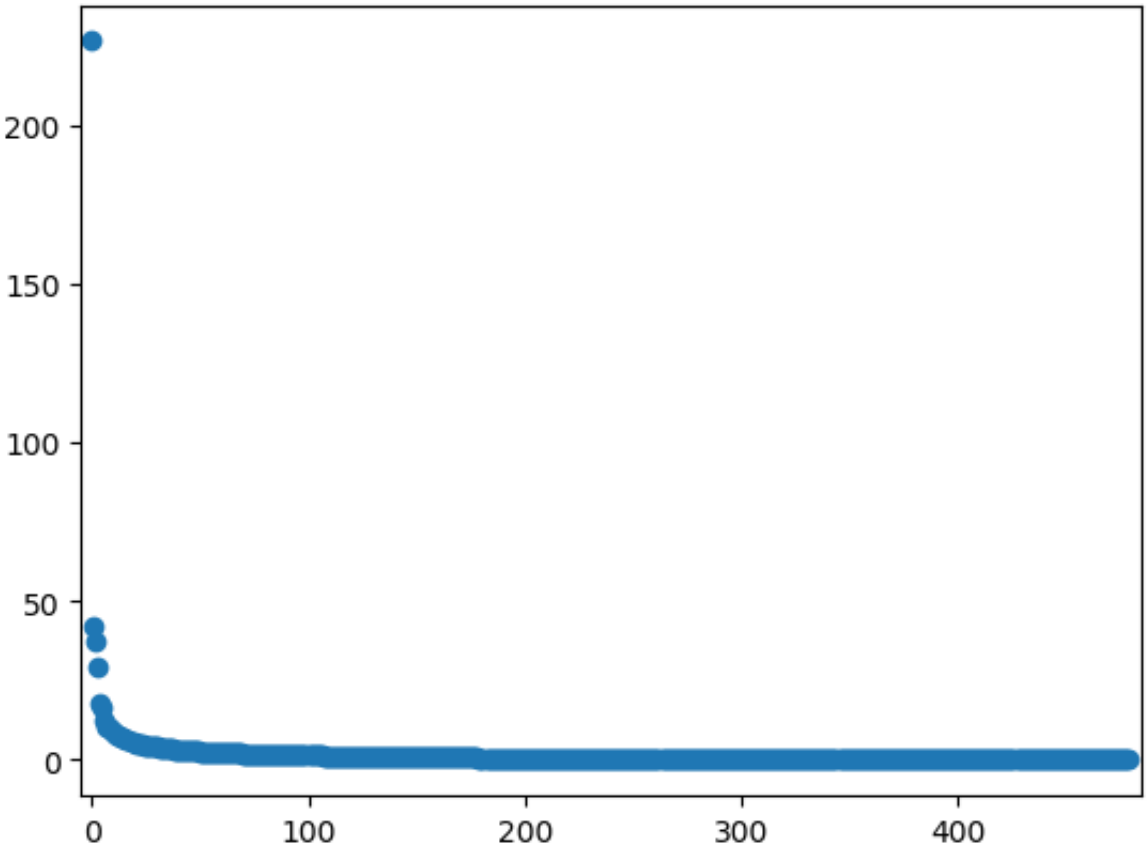


Out[59]: (-5, 485)

In [60]:
```
S_einstein = svdvals(Einstein)
```

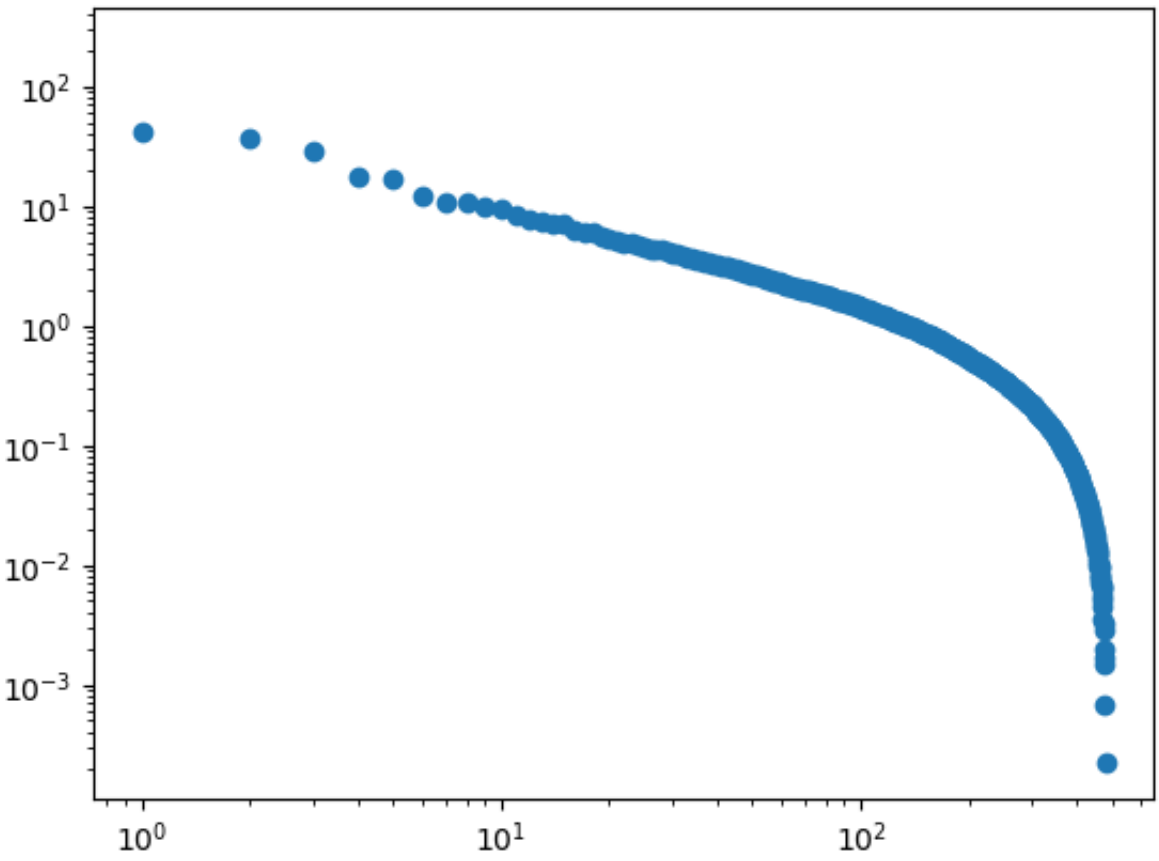
Out[60]:
```
480-element Array{Float32,1}:
 226.28323
  41.76813
  37.076958
  29.329296
  17.734985
  16.62046
  12.27746
  10.7412
  10.579085
   9.776369
   9.407715
   8.38525
   7.594662
   ⋮
   0.006243006
   0.005527602
   0.005114672
   0.0045147766
   0.0035460838
   0.003204308
   0.0028279864
   0.0020040427
   0.0016400132
   0.001459393
   0.00068395055
   0.00022396844
```

In [61]:
```
plot(S_einstein, "o")
xlim(-5, 485)
```



Out[61]: (-5, 485)

In [62]:
```
loglog(S_einstein, "o")
```



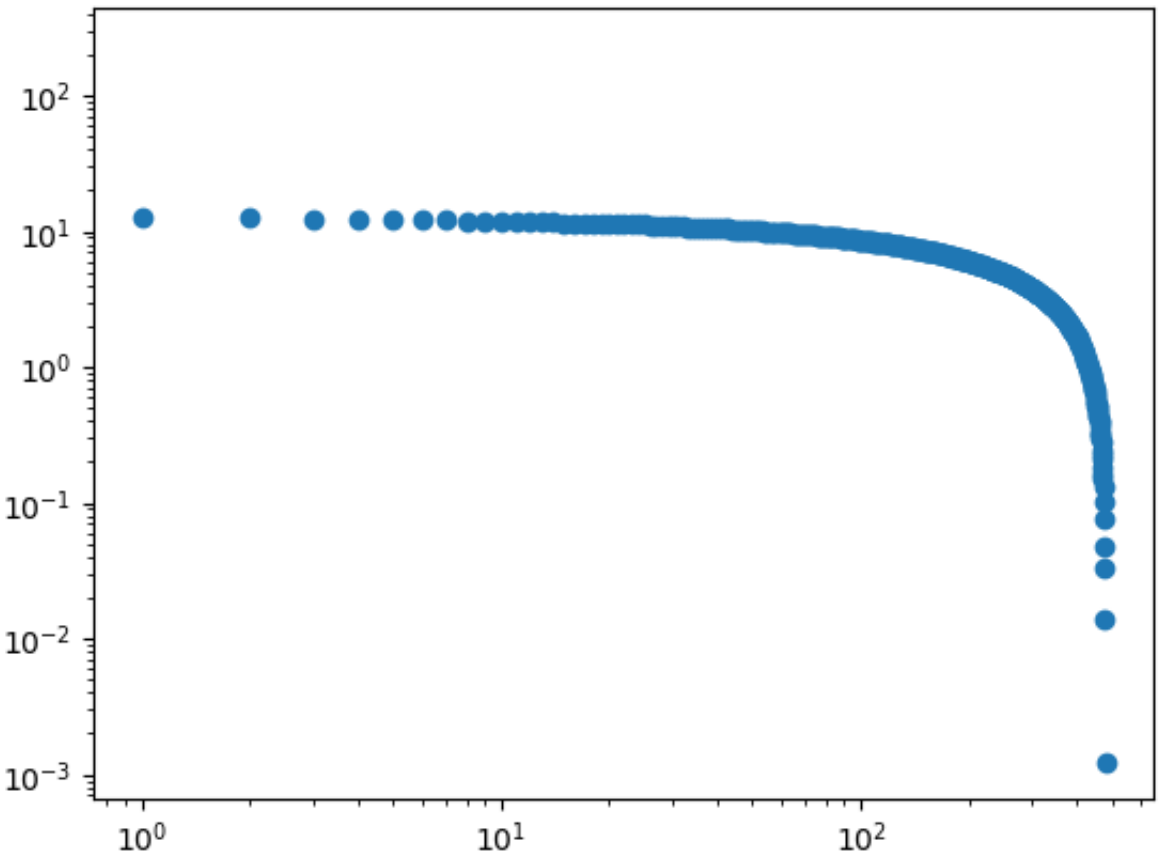
Out[62]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x140f95a58>
```

In [63]:
```
Zufall = rand(480,480);
```

In [64]:
```
S_zufall = svdvals(Zufall)
```

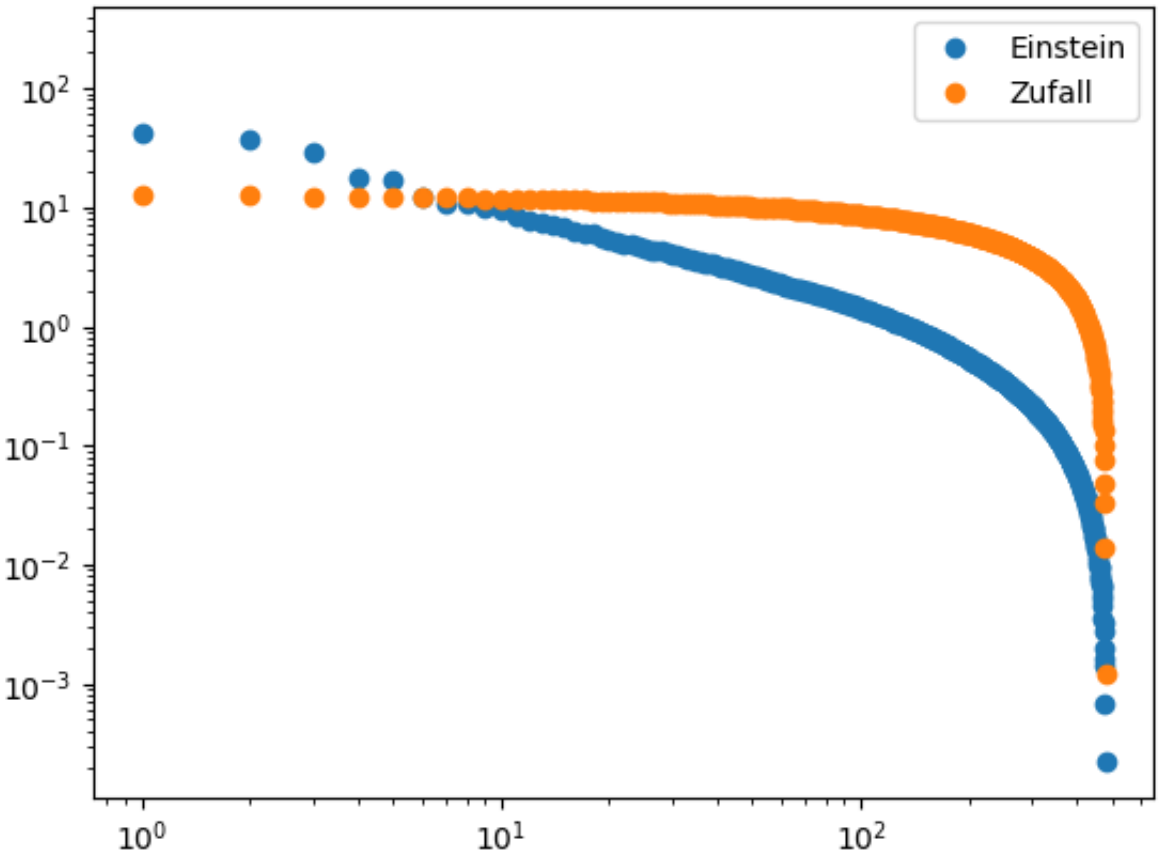
Out[64]:
```
480-element Array{Float64,1}:
 239.98349130076568
  12.63650543968252
  12.50824161899784
  12.415762345063648
  12.325130146438058
  12.116884738717667
  12.107797502380556
  12.03015178773097
  11.971294516954465
  11.901701263599302
  11.814960835721793
  11.762947379680332
  11.7328561507098
   ⋮
   0.2307537484624751
   0.20729801046112886
   0.1839746021018398
   0.16445421532959528
   0.1509524895322564
   0.13227160383444375
   0.10026907806687181
   0.07556452738969394
   0.04738322515666212
   0.03265000897323146
   0.013822658035030761
   0.0012306409485880513
```

In [65]:
```
loglog(S_zufall, "o")
```



Out[65]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x1411713c8>
```

In [66]:
```
loglog(S_einstein, "o", label="Einstein")
loglog(S_zufall, "o", label="Zufall")
legend()
```



Out[66]: PyObject <matplotlib.legend.Legend object at 0x11242d128>

In [ ]: