Elementos de Álgebra Lineal

LES RECOMIENDO checar:

Libros/Cursos en la red:

ALGEBRA LINEAL CON JULIA Stephen Boyd & Lieven Vandenberghe

https://web.stanford.edu/~boyd/vmls/

3BLUE1BROWN EXCELENTE!

https://www.youtube.com/c/3blue1brown

GILBERT STRANG MIT

https://math.mit.edu/g̃s/

Lenguajes de Programación:

Julia

https://julialang.org/

https://github.com/fonsp/Pluto.jl

Anaconda Python

[https://docs.anaconda.com/anaconda/install/(https://docs.anaconda.com/anaconda/install/)

1)Sistemas de Ecuaciones lineales

2)Formulación matricial, Espacios vectoriales,columna,renglón,bases

3)Sistemas determinados, sobre,sub

4)Cuadrados mínimos

5)Problema eigenvectores,eigenvalores

5)Descomposición de valor singular

7)matrices,rotaciones,números complejos

```julia
using Symbolics
```

```julia
 using LinearAlgebra,Latexify, Printf, WGLMakie#,Plots
```

```julia
using LaTeXStrings
```

```julia
using NCDatasets
```

```julia
WGLMakie.activate!()
```

```julia
Latexify.set_default(; starred=true)
```

$[x, y, z]$

```julia
@variables x y z
```

```
round4 (generic function with 1 method)
```

Sistemas de ecuaciones lineales

$$\begin{bmatrix} x + 2y \\ 2x + 4y \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$$

```
begin
A=[1 3 5;4 5 6;7 8 11];
B=[1 2;2 4];
f1=B*[x;y]
f=A*[x; y; z]
#A*[1 2 3]'
d2sys=(f1 ~ [0.0;0.0])
#([[f[1] ~ 2.0,f[2]~0.0,f[3]~0.0]])
#Symbolics.solve_for([f[1] ~ 2.0,f[2]~0.0,f[3]~0.0],[x y z])
end
```

$$\begin{bmatrix} x + 3y + 5z \\ 4x + 5y + 6z \\ 7x + 8y + 11z \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

```
latexify(A*[x;y;z] ~ [2;0;0])
```

$$\begin{bmatrix} x + 3y + 5z \\ 4x + 5y + 6z \\ 7x + 8y + 11z \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}$$

```
(f ~ [4;0;0])
```

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -2.0 \\ 4.8571 \\ -2.7143 \end{bmatrix}$$

```
begin
fsol=Symbolics.solve_for([f[1] ~ -1.0,f[2]~0.0,f[3]~-5.0],[x y z]);
rfsol=[round(fsol[l];digits=4) for l in eachindex(fsol)];
#str1="the solution of this system is [x,y,z]= "
latexify(("[x,y,z]" ~ round4.(rfsol)))
#
#e1=([1; 3; 5])
#e2=([4 5 6]')
#e3=([7 8 11]')
#tt1=latexify(x*e1)
#tt2=latexify(y*e2)
#latexify(x*e1+y*e2+z*e3 ~ [2;0;0])
end
```

El sistema de ecuaciones lo podemos escribir de esta forma

$$x * \vec{col_1}A + y * \vec{col_2}A + z * \vec{col_3}A$$

$$x \cdot \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} + y \cdot \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix} + z \cdot \begin{bmatrix} 5 \\ 6 \\ 11 \end{bmatrix}$$

```
begin

#using Latexify
struct Ket{T}
    x::T
end
@latexrecipe function ff(x::Ket)
return Expr(:latexifymerge, "", x.x, "")
end
latexify(:(x*$(Ket(A[:,1])) + y*$(Ket(A[:,2]))+z*$(Ket(A[:,3]))))
end
```

El sistema de ecuaciones de 2x2

$$\begin{bmatrix} 1.5x + 2.0y \\ 2.0x + 4.0y \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

```
begin
#Symbolics.solve_for([f1[1] ~ 1.0,f1[2] ~ 0.0],[x y])
Bb=copy(B)
Bb=Bb+([0.5 0;0 0])
#f3=B*[x;y];
f3=Bb*[x;y]
#f3=(B+ones(2,2))*[x;y];
(f3~[2; 2])
end
```

```
Symbolics.solve_for([f[1] ~ 2.0,f[2]~0.0,f[3]~0.0],[x y z])
```

La solución del sistem es:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3.0 \\ -1.25 \end{bmatrix}$$

```
begin
fsol3=Symbolics.solve_for([f3[1] ~ 2.0,f3[2] ~ 1.0],[x y])
latexify(("[x,y]" ~ [round4(fsol3[1]);round4(fsol3[2])]))

#t1*string(fsol3)"
#println("The solution is [x,y,z]= " * string(fsol3))
end
```

$$\text{ones}\,(3,3) = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Otros sistemas y su soluciones

1.0

```
@bind mfac1 Slider(-10.0:10.0,show_value=true,default=1)
```

$$M1 = \begin{bmatrix} 1.0 & 3.0 & 5.0 \\ 3.0 & 6.0 & 10.0 \\ 1.0 & 0.0 & 4.5 \end{bmatrix}$$

```
begin
M1=copy(float(A))
M1[2,:]=2.0*M1[1,:]+[1.0,0.0,0.0]*mfac1
M1[3,:]=M1[1,:]-[0.0,0.0,1.0]*(mfac1/2.0)-3.0*[0.0,1.0,0.0]
latexify("M1" ~ M1)
end
```

$$rango = 3$$

```
latexify("rango" ~ rank(M1))
```

$$\begin{bmatrix} x + 3y + 5z \\ x + 2\left(x + 3y + 5z \right) \\ x + 4.5z \end{bmatrix} = \begin{bmatrix} 1.0 \\ 2.0 \\ 1.0 \end{bmatrix}$$

```
begin
M=A*[x;y;z]
#print(M)
M[1]=M[1]
M[2]=2*M[1] +mfac1*x
M[3]=M[1] -mfac1/2*z-3y
#M[1]=M[1] +x+y+z
#M[2]=M[2] -x-3y-2z
#M[3]=M[3] +x+y
latexify(M ~ [1.0;2.0; 1.0])
#Symbolics.solve_for([M[1] ~ 1.0,M[2] ~ 1.0,M[3] ~ 1.0],[x y z])
end

```

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.0 \\ -0.037 \\ 0.2222 \end{bmatrix}$$

```
begin
#fsol4=Symbolics.solve_for([M[1] ~ 1.0,M[2] ~ 2.0,M[3] ~ 1.0],[x y z]);
    if(rank(M1) == 3)
    fsol4=Symbolics.solve_for([M[1] ~ 1.0,M[2] ~ 2.0,M[3] ~ 1.0],[x y z]);
    latexify(("[x,y,z]" ~ round4.(fsol4)))
    else
    latexify("Sistema-Singular-rango-incompleto")
    end
#latexify(("[x,y,z]" ~ ([round4(fsol4[1]);round4(fsol4[2]);round4(fsol4[3])])))
end
```

Operaciones elementales, pivotes, descomposición LU

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 11 \end{bmatrix}$$

```
latexify("A" ~ A)
```

$$\left(P1 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ P1A = \begin{bmatrix} 1 & 3 & 5 \\ 0 & 7 & 14 \\ 7 & 8 & 11 \end{bmatrix},\ P2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 7 & 0 & -1 \end{bmatrix},\ P2P1A = \begin{bmatrix} 1 & 3 & 5 \\ 0 & 7 & 14 \\ 0 & 13 & 24 \end{bmatrix},\ P3 = \right.$$

```julia
begin
#A+[0 0 0;2 -1 0;1 0 -1]*A
#A+[0 0 0;0 0 0;1 0 -1]*A
P1=[1 0 0;4 -1 0;0 0 1]
P2=[1 0 0; 0 1 0;7 0 -1]
P3=[1 0 0; 0 1 0;0 13//7 -1]
PA=P1*A
PPA=P2*PA
PPPA=P3*PPA
latexify("P1" ~ [1 0 0;4 -1 0;0 0 1]),
latexify("P1A" ~ PA),
latexify("P2" ~ [1 0 0; 0 1 0;7 0 -1]),
latexify("P2P1A" ~ PPA),
latexify("P3" ~ [1 0 0; 0 1 0;0 13//7 -1]),
latexify("P1P2P3A" ~ PPPA)
end
#latexify(rank([1 3 5;2 6 10;1 2 0]))
```

1

```julia
@bind indp Slider(1:3,show_value=true,default=1)
```

$$\left(P1 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ P1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

```julia
begin
PS=("P"*string(indp));
PSE=eval(Meta.parse(PS))
PSEI=inv(PSE)
latexify(PS ~ PSE),
if indp < 3
latexify(PS*"^-1" ~ Int.(PSEI))
else
latexify(PS*"^-1" ~ PSEI)
end
#latexify(PS),
#latexify(inv(PS))
end
```

$$\left(A = \begin{bmatrix} 1 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 11 \end{bmatrix}, SP = \begin{bmatrix} 1 & 0 & 0 \\ 4 & -1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, SPA = \begin{bmatrix} 1 & 3 & 5 \\ -7 & -1 & 3 \\ 7 & 8 & 11 \end{bmatrix}, 4A\left[1,:\right] - A\left[2,:\right] - A\left[3,:\right] = [\right.$$

```
begin
latexify("A" ~ A),
latexify("SP" ~ [1 0 0;4 -1 -1;0 0 1]),
latexify("SPA" ~ [1 0 0;4 -1 -1;0 0 1]*A),
latexify("4A[1,:]-A[2,:]-A[3,:]" ~ [4*A[1,:]-A[2,:]-A[3,:]]')
end

```

```
3×3 Matrix{Float64}:
 1.0   0.0  -0.0
 4.0  -1.0  -1.0
 0.0   0.0   1.0
```

```
inv([1 0 0;4 -1 -1;0 0 1])
```

Descompoisición LU, A LU?

$$\left(L = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.1429 & 1.0 & 0.0 \\ 0.5714 & 0.2308 & 1.0 \end{bmatrix}, U = \begin{bmatrix} 7.0 & 8.0 & 11.0 \\ 0.0 & 1.8571 & 3.4286 \\ 0.0 & 0.0 & -1.0769 \end{bmatrix}, LU = \begin{bmatrix} 7.0 & 8.0 & 11.0 \\ 1.0 & 3.0 & 5.0 \\ 4.0 & 5.0 & 6.0 \end{bmatrix}\right)$$

```
begin
FLU=lu(A)
latexify(("L " ~round.(FLU.L,digits=4))),
latexify(("U " ~round.(FLU.U,digits=4))),
latexify(("LU " ~round.(FLU.L*FLU.U,digits=4)))
end
```

Hay una permutación de renglones de por medio en el proceso!

$$PA = LU$$

```
latexify("PA = LU")
```

$$\left(A = \begin{bmatrix} 1 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 11 \end{bmatrix}, P = \begin{bmatrix} 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}, PA = \begin{bmatrix} 7 & 8 & 11 \\ 1 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}\right)$$

$$P^{-1}LU = \begin{bmatrix} 1.0 & 3.0 & 5.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 11.0 \end{bmatrix}$$

```
latexify(("(P^-1)LU" ~ inv(FLU.P)*FLU.L*FLU.U))
#latexify(A[sort(FLU.p),:])
```

permuta renglones 1-> 2, 2 -> 3, 3 -> 1

$$P \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

```
#latexify(inv(FLU.P)*FLU.p)
latexify(("P*[0;0;1]"~ FLU.P*[0;0;1]))
```

$$P \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \end{bmatrix}$$

```
latexify(("P*[1;0;0]"~ FLU.P*[1;0;0]))
```

$$P = \begin{bmatrix} 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \end{bmatrix}$$

```
latexify(("P"~ FLU.P))
```

$$U = L^{-1}PA = \begin{bmatrix} 7.0 & 8.0 & 11.0 \\ 0.0 & 1.8571 & 3.4286 \\ 0.0 & 0.0 & -1.0769 \end{bmatrix}$$

```
latexify(("U=(L^-1)PA" ~ round.(inv(FLU.L)*FLU.P*A,digits=4)))
```

$$PA = LU, PAx = Pb, LUx = Pb, Ux = c, Lc = Pb$$

```
latexify("PA=LU, PAx=Pb, LUx=Pb,Ux=c, Lc=Pb")
```

Resuelve para c (sistema Lc=Pb) y luego resuelve Ux=c para obtener x

```
md" ##### Resuelve para c (sistema Lc=Pb ) y luego resuelve Ux=c para obtener x"
```

$$c = L^{-1}P\begin{bmatrix}2\\0\\0\end{bmatrix} = \begin{bmatrix}0.0\\2.0\\-0.4615\end{bmatrix}$$

```
latexify(("c=(L^-1)P*[2;0;0]" ~ round4.(inv(FLU.L)*FLU.P*[2;0;0])))
```

```
[0.0, 2.0, -0.461538]
```

```
begin
#FLU.U
#FLU.U*[x;y;z]#=
invlb=inv(FLU.L)*FLU.P*[2;0;0];
#FLU.p
end
```

$$(U \cdot \begin{bmatrix}x\\y\\z\end{bmatrix} = c, \begin{bmatrix}7.0x + 8.0y + 11.0z\\1.8571y + 3.4286z\\-1.0769z\end{bmatrix} = \begin{bmatrix}0.0\\2.0\\-0.462\end{bmatrix}, \begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}-1.0\\0.2857\\0.4286\end{bmatrix})$$

```
begin
Ux=FLU.U*[x;y;z]
fsol5=Symbolics.solve_for([Ux[1] ~ invlb[1],Ux[2] ~ invlb[2],Ux[3] ~ invlb[3]],[x y z])
latexify("U*[x,y,z] = c"),
latexify(round4.(FLU.U)*[x;y;z] ~ round.(invlb,digits=3)),
latexify([x, y, z] ~ (round4.(fsol5)))
#latexify([round.(FLU.U,digits=3)(:($(Ket([x; y; z])))) ~ round.(invlb,digits=3) ,[x, y, z] ~ (round.(fsol5,digits=4))])
#latexify(round.(FLU.U,digits=4).*[x; y; z])
end
```

La solución 'back slash' [x;y;z]=A\b

```
md"# La solución 'back slash' [x;y;z]=A\b"
```

begin

Definimos la función

function lfo(A,b)

return A\b

end

[x y, z]=lf0(A,[b1;b2;b3])

end

```
md"""

##### begin

###### Definimos la función

function lf0(A,b) \
return A\b \
end

##### [x y, z]=lf0(A,[b1;b2;b3])

##### end
"""
```

```
Enter cell code...
```

$$\left(\begin{bmatrix} 1 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 11 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b1 \\ b2 \\ b3 \end{bmatrix}, b = \begin{bmatrix} 2.0 \\ 0.0 \\ 0.0 \end{bmatrix}, \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1.0 \\ 0.2857 \\ 0.4286 \end{bmatrix} \right)$$

```julia
#Defined above remember!
#using Latexify
#struct Ket{T}
#    x::T
#end
#@latexrecipe function f(x::Ket)
#return Expr(:latexifymerge, "", x.x, "")
#end

begin
lf0(A,b)=A\b
xsol0=lf0(A,[2.0;0.0;0.0])
#strAb=(A\b)
b3=[2.0;0.0;0.0]
#latexify(:(x*$(Ket(A[:,1])) + y*$(Ket(A[:,2]))+z*$(Ket(A[:,3])))),
latexify(:($(Ket(A[:,:]))*[x;y;z] = [b1;b2;b3])),#*[x;y;z]), #+
y*$(Ket(A[:,2]))+z*$(Ket(A[:,3])))),
latexify("b" ~ b3),
#latexify(strAb)
latexify(("[x,y,z]" ~ round.(xsol0,digits=4)))
end
```

Cambia el lado derecho de la ecuación y resuelve x=A\b

$$b = \begin{bmatrix} -2.0 \\ 1.0 \\ -2.0 \end{bmatrix}$$

```julia
latexify("b" ~ [-2,1.0,-2.0] )
```

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1.5 \\ -0.571 \\ -0.357 \end{bmatrix}$$

```julia
begin
lf(A,b)=A\b
xsol1=lf(A,[-2.0;1.0;2.0])
latexify(("[x,y,z]" ~ round.(xsol1,digits=3)))

end
```

```
Point{2, Float32}
```

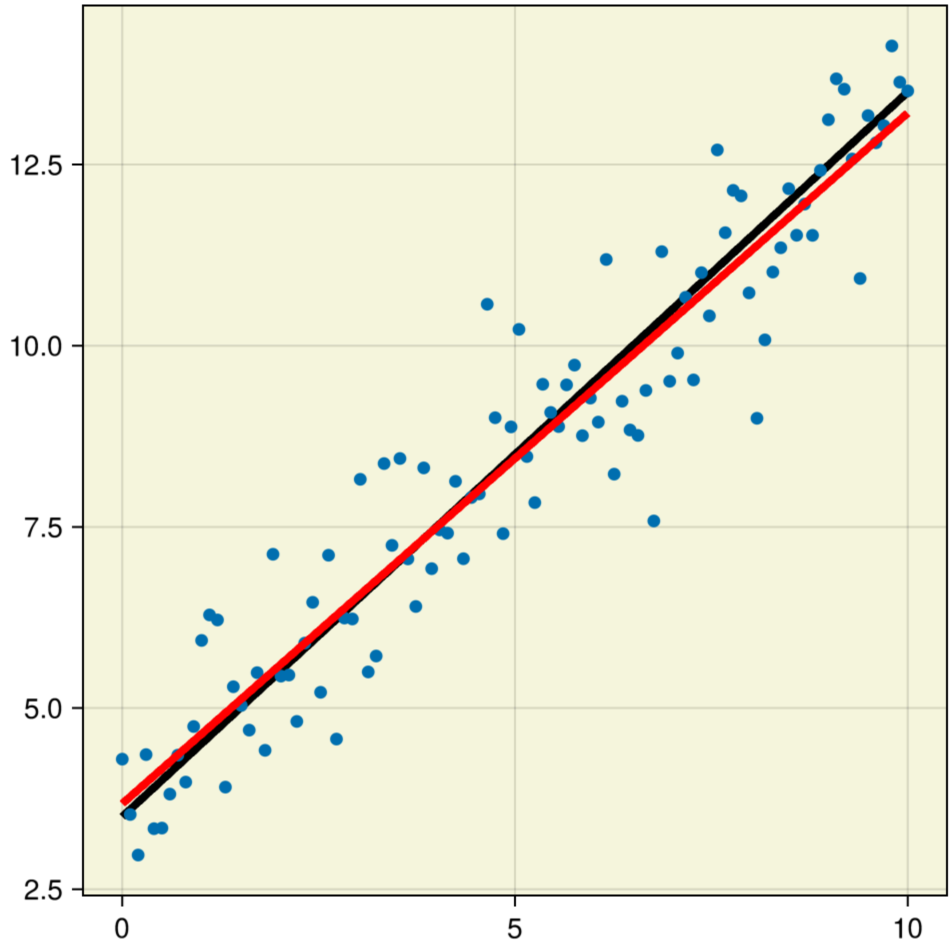
```julia
Point2f
```

Cuadrados mínimos, caso típico de problema sobre-determinado

```
md"## Cuadrados mínimos, caso típico de problema sobre-determinado"
```

1.0

```
@bind inoise Slider(0.0:0.2:5,show_value=true,default=1)
```




```
begin
flsq = Figure(size = (500, 500))
axl=Axis(flsq[1, 1], backgroundcolor = "beige")
tt=LinRange(0,10,100);
#tt=0:.1:10.0;
noise=inoise*randn(length(tt));
Att=[tt[:] ones(length(tt),1)];
xv=[1.0,3.5];
xlsq=lf0(Att,Att*xv+noise);
yobs=Att*xv.+noise
yv=Att*xv
ylsq=Att*xlsq
pointsobs = Point2f.(tt, yobs)
pointsv=Point2f.(tt, yv)
lines!(axl,pointsv,linewidth=4.0,color=:black)
scatter!(axl,pointsobs)
lines!(axl,tt,ylsq,linewidth=4.0,color=:red)
#scatter(tt,Att*xv+noise)#plot(tt,Att*xlsq,color=:red)
#lines(tt,yv)
#xlsq=lf0(Att,Att*xv+noise);
#size(noise)
#size(Att*xv)
#limits!(axl,0.0,10.0,0.0,16.0)
flsq
end

```

[0.95297, 3.68057]

```
xlsq
```

$$\left(xlsq = \begin{bmatrix} 0.9529703518435175 \\ 3.680571332535638 \end{bmatrix},\ xv = \begin{bmatrix} 1.0 \\ 3.5 \end{bmatrix}\right)$$

```
begin
latexify("xlsq" ~ xlsq),
latexify("xv" ~ xv)
end
```

$A^t(A\vec{x} -b)= 0$

$A^tA\vec{x} = A^tb$

$\vec{x} = (A^tA)^{-1}A^tb$

Las diferencias (residuos) deben ser ortogonales al espacio columna de A

```
md"""
#### $A^t(A\vec{x} -b)= 0$
#### $A^tA\vec{x} = A^tb$
#### $\vec{x} = (A^tA)^{-1}A^tb$
##### Las diferencias (residuos) deben ser ortogonales al espacio columna de A
"""
```

$$\left[A^t(A\vec{x} - b),\ =,\ \begin{bmatrix} -3.1796787425264483e-13 \\ -4.1300296516055823e-14 \end{bmatrix}\right]$$

```
begin
slsq=L"{A^t(A\vec{x} -b)}"
residual= Att'*(Att*xlsq-yobs)
latexify(slsq ~ residual)
([latexify(slsq); latexify(L"="); latexify(residual)])
end
```

$$\$A^t(A\vec{x} - b)\$ = \begin{bmatrix} -3.1796787425264483e-13 \\ -4.1300296516055823e-14 \end{bmatrix}$$

```
begin
slsq1=L"{C^t(C\vec{x} -d)}"
latexify(slsq ~ residual)
end
```

Eigenvalores/Eigenvectores (Matrices cuadradas)

$$A * \vec{v}_j = \lambda_j \vec{v}_j$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

$$AV = [A\vec{v}_1 \quad A\vec{v}_2 \quad \ldots \quad A\vec{v}_n] = [\lambda_1\vec{v}_1 \quad \lambda_2\vec{v}_2 \quad \ldots \quad \lambda_n\vec{v}_n]$$

$$A * V = V * \Lambda$$

$$V^{-1} * A * V = \Lambda$$

$$V * \Lambda * V^{-1} = A$$

Descomposición en Valores Singulares (SVD)

$$A^{m\times n} = U^{m\times m} S^{m\times n} V^{T(n\times n)}$$

$$U^T * U = I^{m\times m}, V^T * V = I^{n\times n}$$

U, V tienen columnas ortonormales

```
g (generic function with 1 method)
```

```
begin
@variables a[1:2,1:2]
function g(a,c)
    return a=c
end
end
```

$$aa = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

```
latexify("aa" ~ aa)
```

$$\begin{bmatrix} -0.618 \\ 1.618 \end{bmatrix}$$

```
latexify(round4.(L1))
```

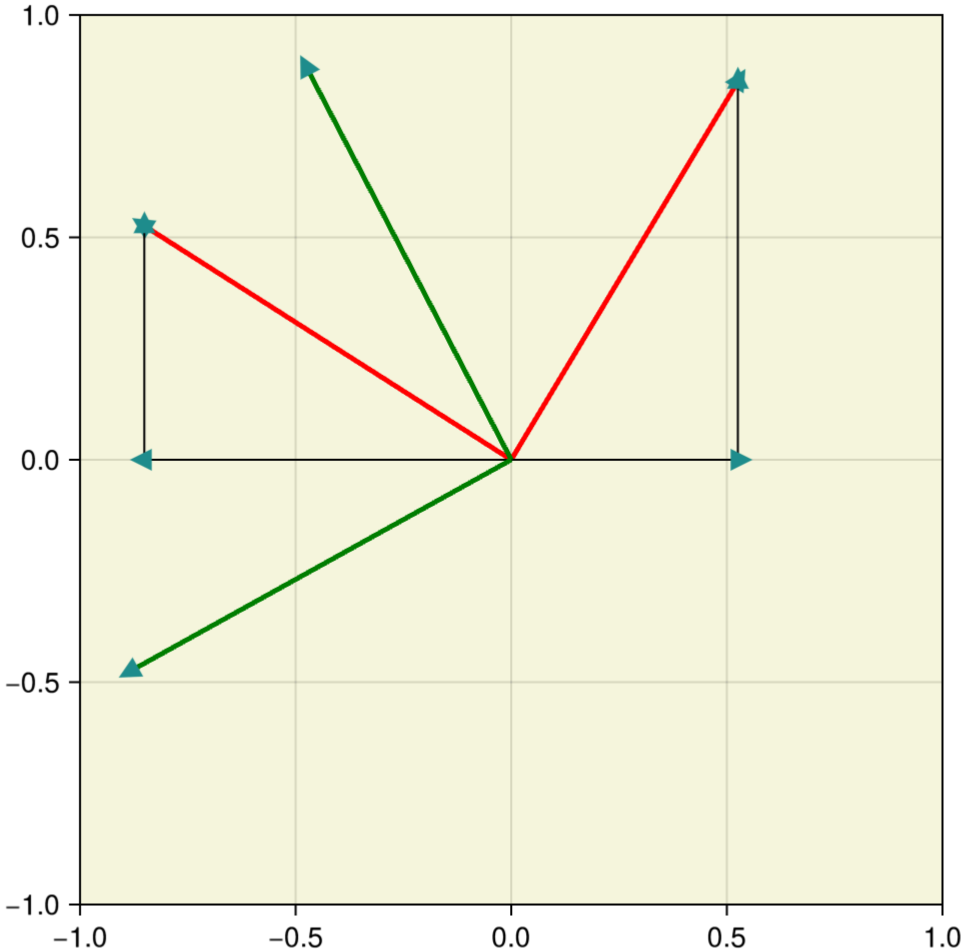
$$L = V^{-1}\mathrm{aa}\left(V \right) = \begin{bmatrix} -0.618 & -0.0 \\ -0.0 & 1.618 \end{bmatrix}$$

```
begin
    aa=g(a,[0 1;1 1]);
L1,V1=eigen(aa)
    latexify(["L = (V^-1)aa(V)" ~ round.(inv(V1)*aa*(V1),digits=4)])
#   V1*diagm(L1)*inv(V1)
end

```

$$eigvecs = \begin{bmatrix} -0.8507 & 0.5257 \\ 0.5257 & 0.8507 \end{bmatrix}$$

```
latexify("eigvecs" ~ round4.(V1))
```




```
begin
ff = Figure(size = (500, 500))
ax=Axis(ff[1, 1], backgroundcolor = "beige")
xs = LinRange(-10, 10, 21)
ys = LinRange(-10, 10, 21)
us = [1.0 for x in xs, y in ys]
vs = [0.0 for x in xs, y in ys]
strength = vec(sqrt.(us .^ 2 .+ vs .^ 2))
#arrows!(xs, ys, us, vs, arrowsize = 10, lengthscale = 1.0, arrowcolor = strength,
linecolor = :yellow)
#arrows!(xs, ys, vs, us, arrowsize = 10, lengthscale = 1.0, arrowcolor = strength,
linecolor = :yellow)
strength2 = vec(sqrt.(V1[:,1] .^ 2 .+ V1[:,2].^ 2))
arrows!(ax,[0;0], [0;0], V1[1,:], V1[2,:],arrowsize =15.0 , lengthscale
=1.,arrowcolor = strength2, linecolor =:red,linewidth=2.5)
#arrows!(ax,[0;0], [0;0], [0;1], [1;0],arrowsize =15.0 , lengthscale =1.,arrowcolor =
[1;1], linecolor =:black)
arrows!(ax,[0;V1[1,1]],[0;0],[V1[1,1],0],[0,V1[2,1]],arrowsize =15.0 , lengthscale
=1.,arrowcolor =[1;1], linecolor =:black)
arrows!(ax,[0;V1[1,2]],[0;0],[V1[1,2],0],[0,V1[2,2]],arrowsize =15.0 , lengthscale
=1.,arrowcolor =[1;1], linecolor =:black)
θ=-atan(V1[2,2]/V1[1,2]);
θ=π/3
r90=[cos(θ) -sin(θ);sin(θ) cos(θ)];
V2=r90*V1;

```

```
arrows!(ax,[0;0], [0;0], V2[1,:], V2[2,:],arrowsize =15.0 , lengthscale
=1.,arrowcolor = strength2, linecolor =:green,linewidth=2.5)
#arrows!(ax,[0;V1[1,1]],[0;0],[V1[1,1],0],[0,V1[2,1]],arrowsize =15.0 , lengthscale
=1.,arrowcolor =[1;1], linecolor =:black)
#arrows!(ax,[0;0], [0;0], [0;1], [1;0],arrowsize =15.0 , lengthscale =1.,arrowcolor =
[1;1], linecolor =:black)
limits!(ax, -1, 1, -1, 1)
ff
```

$$\left(A = \begin{bmatrix} 1 & 3 & 5 \\ 4 & 5 & 6 \\ 7 & 8 & 11 \end{bmatrix}, eigvalsA = \begin{bmatrix} -1.7177 \\ 0.4461 \\ 18.2716 \end{bmatrix}, eigvecsA = \begin{bmatrix} 0.9064 & 0.2413 & 0.3198 \\ -0.2148 & -0.8437 & 0.4687 \\ -0.3638 & 0.4795 & 0.8235 \end{bmatrix}\right)$$

```
begin
LA,VA=eigen(A);
latexify("A" ~ A),
latexify("eigvalsA" ~ round4.(LA)),
latexify("eigvecsA" ~ round4.(VA))
end
```

```
cc = 2×2 Matrix{Int64}:
      2  3
      4  6
```

```
cc=[2 3;4 6]
```

kk =

$$\begin{bmatrix} 3.0x + 4.0y \\ 5.0x + 7.0y \end{bmatrix}$$

```
kk=(cc+ ones(2,2))*[x y]'
```

```
[1.0, -0.5]
```

```
Symbolics.solve_for([kk[1] ~ 1,kk[2] ~ 1.5],[x y])
```

```
0
```

```
Uc,Sc,Vc=svd(cc);
```

```
2×2 Matrix{Int64}:
 2  3
 4  6
```

```
#-.5*cc[:,1]
cc
```

xx (generic function with 1 method)

```
begin
function xx(yy)
return [-1.5yy,yy]
end
#xx[1.0]
end
```

[[15.0, -10.0], [13.5, -9.0], [12.0, -8.0], [10.5, -7.0], [9.0, -6.0], [7.5, -5.0], [6.0, -

```
begin
ff1 = Figure(size = (800, 800))
ax1=Axis(ff1[1, 1], backgroundcolor = "black")
#for l=-10:10
l=LinRange(-10, 10, 21)
xn=xx.(l)
#scatter!(ax1,xn[1],xn[2])#,xn[1]*cc[:,1]+xn[2]*cc[:,2])
#end

end
```

[-10.0, -9.0, -8.0, -7.0, -6.0, -5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0,

```
begin
x1s=[xn[l1][1] for l1=1:length(l)];
y1s=[xn[l1][2] for l1=1:length(l)];
end
```

Combinaciones lineales de las columnas de la matriz (lin indep) me dan cualquier vector

```
md"Combinaciones lineales de las columnas de la matriz (lin indep) me dan cualquier
vector"
```

$$cc = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix}$$

```
latexify("cc"~ cc)
```

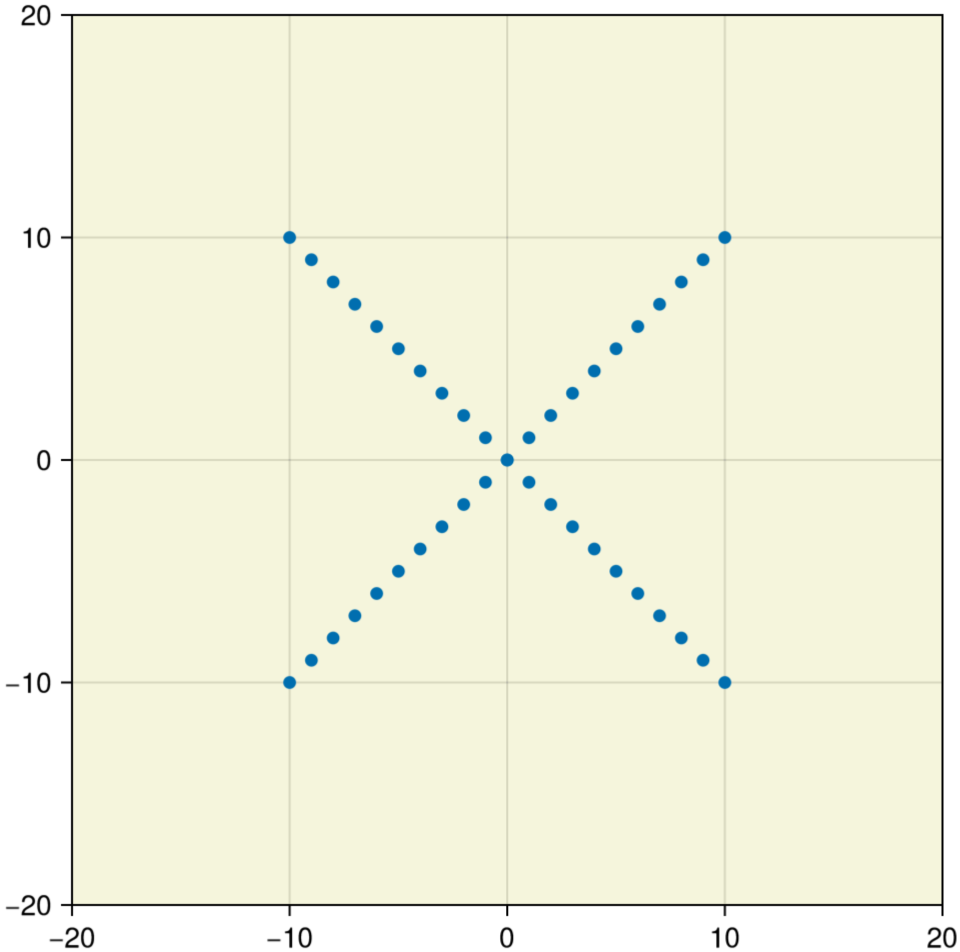
Vector $[50, 10]$ lo puedo escribir como

```
md"Vector $[50,10]$ lo puedo escribir como"
```

$$\begin{bmatrix} 50.0 \\ 100.0 \end{bmatrix}$$

```
latexify(10*cc[:,1]+10*cc[:,2] -100*cc[:,1] + 200/3*cc[:,2])
```

```
factor=[-5:5];
```




```
begin
ff0 = Figure(size= (500, 500))
ax0=Axis(ff0[1, 1], backgroundcolor = "beige")
#scatter(xs,ys)
#scatter(2l,4l)
scatter!(vec([xs ys]),vec([-ys xs]))
limits!(-20,20,-20,20)
ff0
end
```

```
Enter cell code...
```

$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & -1 \\ 1 & -2 & 3 \end{bmatrix}$$

descomposición QR, A=QR

Q, Orthogonal, R upper triangular

```
md""" ## descomposición QR, A=QR
### Q, Orthogonal, R upper triangular
"""
```

$$(R = \begin{bmatrix} -1.7321 & 1.7321 & -1.7321 \\ 0.0 & -1.4142 & 2.8284 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}, Q = \begin{bmatrix} -0.5774 & -0.7071 & 0.4082 \\ 0.5774 & 0.0 & 0.8165 \\ -0.5774 & 0.7071 & 0.4082 \end{bmatrix})$$

```
begin
q,r=qr(BB);
latexify("R " ~ round4.(r)),
latexify("Q " ~ round4.(q))
end
```

$$(QR = \begin{bmatrix} 1.0 & -0.0 & -1.0 \\ -1.0 & 1.0 & -1.0 \\ 1.0 & -2.0 & 3.0 \end{bmatrix}, BB = \begin{bmatrix} 1.0 & 0.0 & -1.0 \\ -1.0 & 1.0 & -1.0 \\ 1.0 & -2.0 & 3.0 \end{bmatrix})$$

```
begin
latexify("QR" ~ round4.(q*r)),
latexify("BB" ~ round4.(BB))
end
```

$$(E1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, E2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, E3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \end{bmatrix})$$

```
begin
E1=[1 0 0;1 1 0;0 0 1]
E2=[1 0 0;0 1 0;-1 0 1]
E3=[1 0 0;1 0 0;0 2 1]
latexify("E1" ~ E1),
latexify("E2" ~ E2),
latexify("E3" ~ E3)
end
```

$$\left(BB = \begin{bmatrix} 1.0 & 0.0 & -1.0 \\ -1.0 & 1.0 & -1.0 \\ 1.0 & -2.0 & 3.0 \end{bmatrix}, E1BB = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 1 & -2 & 3 \end{bmatrix}, E2E1BB = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & -2 & 4 \end{bmatrix}, E3E2E1BB\right.$$

```
begin
latexify("BB" ~ round4.(BB)),
latexify("E1*BB" ~ E1*BB),
latexify("E2*E1*BB" ~ E2*E1*BB),
latexify("E3*E2*E1*BB" ~ round4.(E3*E2*E1*BB))
end
```

[-2 -4 2]

```
@bind d1 PlutoUI.Slider([[i  2i -i] for i in -2:2],show_value=true,default=1)
```

[-3.0 -8.0 -2.0]

```
@bind  d2 PlutoUI.Slider([[3j/2 4j j] for j in -2:2],show_value=true,default=1)#,
for k in -2:2])
```

[-6.0 0.4 -8.0]

```
@bind  d3 PlutoUI.Slider([[3k -k/5 4k] for k in -2:2],show_value=true,default=1)
```

$$\begin{bmatrix} -2.0 & -4.0 & 2.0 \\ -3.0 & -8.0 & -2.0 \\ -6.0 & 0.4 & -8.0 \end{bmatrix}$$

```
begin
Md=[d1;d2;d3]
    latexify(Md)
end
```

```
[-0.9467, 0.1444, 0.3422]
```

```
begin
lf(Md,[2;1;3])
round.(lf(Md,[2;1;3]),digits=4)
end
```

$$L = \begin{bmatrix} -10.2967 + 0.0i & 0.0i & 0.0i \\ 0.0i & -3.8517 - 1.6267i & 0.0i \\ 0.0i & 0.0i & -3.8517 + 1.6267i \end{bmatrix}$$

```
begin
EM=eigen(Md)
    EM.values
latexify("L" ~ diagm(round4.(EM.values)))
end
```

$$V = \begin{bmatrix} -0.2586 + 0.0i & 0.5478 - 0.2079i & 0.5478 + 0.2079i \\ -0.804 + 0.0i & -0.0609 + 0.1265i & -0.0609 - 0.1265i \\ -0.5354 + 0.0i & -0.7981 + 0.0i & -0.7981 + 0.0i \end{bmatrix}$$

```
#EM.vectors
latexify("V" ~ (round4.(EM.vectors)))
```

```
3×1 Matrix{ComplexF64}:
 8.2743 + 6.0224im
 1.9914 + 8.493im
 7.3066 + 0.317im
```

```
round4.((rand(3,1)+im*rand(3,1))*10)
```

```
LinearProblem. In-place: true
b: 3×1 Matrix{Float64}:
 15.117
 16.919
  1.58
```

```
begin
brhs=round.(rand(3,1)*20,digits=3)
Problema=LinearProblem(Md,brhs)
#sol = solve(Problema,IterativeSolversJL_GMRES())
#sol=solve(Problema)
#round.(sol.u,digits=3)
end
```

$$xt = \begin{bmatrix} 4.9458 \\ -0.0033 \end{bmatrix}$$
$$xt2 = \begin{bmatrix} 4.9458 \\ -0.0033 \end{bmatrix}$$
$$xt3 = \begin{bmatrix} 4.9458 \\ -0.0033 \end{bmatrix}$$
$$xt4 = \begin{bmatrix} 4.9458 \\ -0.0033 \end{bmatrix}$$

```
begin
t=LinRange(0.0,100,200);
At=[ones(200,1) t];
bt=rand(200,1)*10.0;
xt=inv(transpose(At)*At)*transpose(At)*bt
xt2=At\bt
xt3=pinv(At)*bt
typeof(vec(bt))
(xt4,stats) = cgls(At,vec(bt))
latexify(["xt" ~ round4.(xt), "xt2" ~ round4.(xt3),"xt3" ~ round4.(xt3), "xt4" ~
round4.(xt4)])
end
```

$$\left(\begin{bmatrix} -0.2586+0.0i & 0.5478-0.2079i & 0.5478+0.2079i \\ -0.804+0.0i & -0.0609+0.1265i & -0.0609-0.1265i \\ -0.5354+0.0i & -0.7981+0.0i & -0.7981+0.0i \end{bmatrix}, \begin{bmatrix} -0.2586+0.0i & -0.804+0.0i \\ 0.5478+0.2079i & -0.0609-0.126 \\ 0.5478-0.2079i & -0.0609+0.126 \end{bmatrix} \right.$$

```julia
latexify(round4.(EM.vectors)),
latexify(round4.(EM.vectors')),
latexify(round4.(conj.(EM.vectors)))
```

```
EMV = Eigen{ComplexF64, Float64, Matrix{ComplexF64}, Vector{Float64}}
      values:
      3-element Vector{Float64}:
       -3.8596953069823394
        0.8685009737929961
       36.99119433318928
      vectors:
      3×3 Matrix{ComplexF64}:
       -0.905599-0.00361395im   -0.16496-0.118296im    -0.365486+0.0713593im
       0.0510555+0.125175im     0.852368+0.00319519im  -0.501256+0.0626551im
        0.401998-0.0im         -0.481926-0.0im         -0.778553-0.0im
```

```julia
EMV=eigen((A+im*A)+(A+im*A)')
```

$$\left(\begin{bmatrix} 2.0+0.0i & 7.0-1.0i & 12.0-2.0i \\ 7.0+1.0i & 10.0+0.0i & 14.0-2.0i \\ 12.0+2.0i & 14.0+2.0i & 22.0+0.0i \end{bmatrix}, \begin{bmatrix} -0.9056-0.0036i & -0.165-0.1183i & -0.3655+0.07 \\ 0.0511+0.1252i & 0.8524+0.0032i & -0.5013+0.06 \\ 0.402+0.0i & -0.4819+0.0i & -0.7786+0. \end{bmatrix} \right.$$

```julia
begin
latexify(round4.((A+im*A)+(A+im*A)')),
latexify(round4.(EMV.vectors)),
latexify(round4.(EMV.values))
end
```

$$\left(\begin{bmatrix} 2.0+0.0i & 7.0+1.0i & 12.0+2.0i \\ 7.0-1.0i & 10.0+0.0i & 14.0+2.0i \\ 12.0-2.0i & 14.0-2.0i & 22.0+0.0i \end{bmatrix}, \begin{bmatrix} -0.9056+0.0036i & -0.165+0.1183i & -0.3655-0.07 \\ 0.0511-0.1252i & 0.8524-0.0032i & -0.5013-0.06 \\ 0.402+0.0i & -0.4819+0.0i & -0.7786+0. \end{bmatrix} \right.$$

```julia
begin
EMVC=eigen(conj.((A+im*A)+(A+im*A)'))
latexify(round4.(conj.((A+im*A)+(A+im*A)'))),
latexify(round4.(EMVC.vectors)),
latexify(round4.(EMVC.values))
end
```

$$\left(\begin{bmatrix} -2.0i & -1.0-7.0i & -2.0-12.0i \\ 1.0-7.0i & -10.0i & -2.0-14.0i \\ 2.0-12.0i & 2.0-14.0i & -22.0i \end{bmatrix}, \begin{bmatrix} 0.9056+0.0i & 0.3655-0.0714i & -0.1654-0.1 \\ -0.0516-0.125i & 0.5013-0.0627i & 0.8524+0.0 \\ -0.402+0.0016i & 0.7786+0.0i & -0.4819+0.0 \end{bmatrix}\right.$$

```
begin
EMVD=eigen(conj.((A+im*A)-(A+im*A)'))
latexify(round4.(conj.((A+im*A)-(A+im*A)'))),
latexify(round4.(EMVD.vectors)),
latexify(round4.(EMVD.values))
end
```

$(1+5i,\ 1+5i,\ 1-5i)$

```
begin
p11=[1+im; 2+3*im]
p12=[5-im;-3 + im]
latexify(p11'* p12),
latexify(conj(p12'*p11)),
latexify(p12'*p11)
end

```

```
pascal (generic function with 1 method)
```

```
pascal(N) = [binomial(n, k) for n = 0:N, k=0:N]
```

```
Pas = 6×6 Matrix{Int64}:
       1  0   0   0  0  0
       1  1   0   0  0  0
       1  2   1   0  0  0
       1  3   3   1  0  0
       1  4   6   4  1  0
       1  5  10  10  5  1
```

```
Pas=pascal(5)
```

```
6×6 Matrix{Int64}:
 0  -1  -1  -1  -1   -1
 1   0  -2  -3  -4   -5
 1   2   0  -3  -6  -10
 1   3   3   0  -4  -10
 1   4   6   4   0   -5
 1   5  10  10   5    0
```

```
Pas-Pas'
```

$$eigenvalues - pasc - pasc' = \begin{bmatrix} -18.5547i \\ 18.5547i \\ -0.1954i \\ 0.1954i \\ -0.8277i \\ 0.8277i \end{bmatrix}$$

```
begin
PasE=eigen(Pas-Pas')
@printf "%s " "eigenvalues pasc - pasc^T "
latexify(["eigenvalues-pasc-pasc'" ~ round4.(PasE.values)])
end
```

```
eigenvalues pasc - pasc^T
```

```
rotating_matrices3d (generic function with 1 method)
```

```
begin
    function rotating_matrices3d(α,β,θ)
        matz=[cos(θ) -sin(θ) 0.0; sin(θ) cos(θ) 0.0; 0.0 0.0 1.0];
        matx=[1.0 0.0 0.0;0.0 cos(α) -sin(α); 0.0 sin(α) cos(α)];
        maty=[cos(β) 0.0 sin(β); 0.0 1.0 0.0; -sin(β) 0.0 cos(β)];
        return matx,maty,matz
    end
end

```

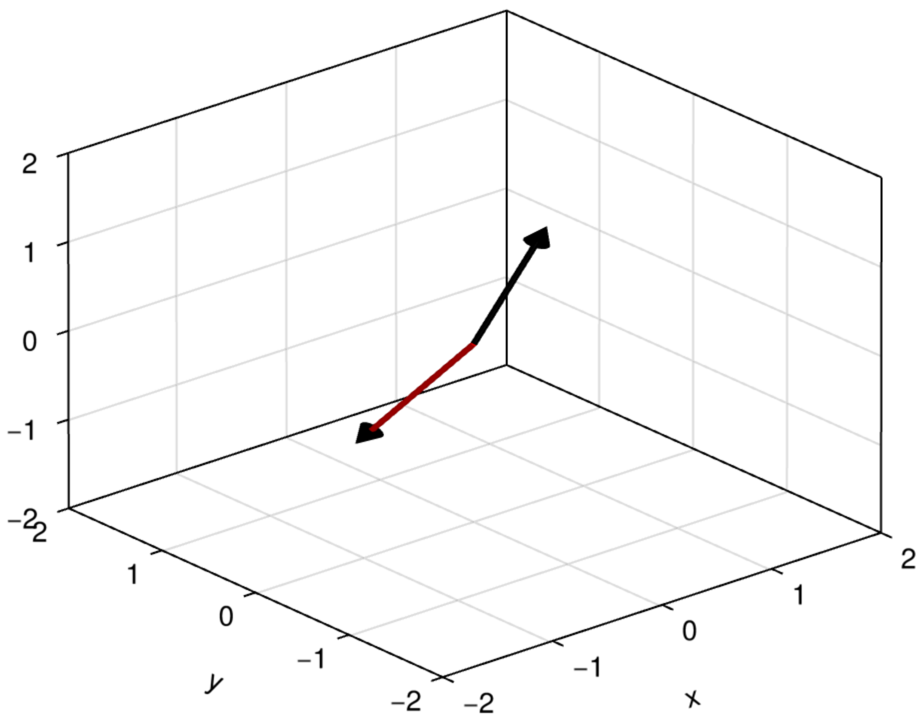
```
mat3d=rotating_matrices3d(0.0,π/2,π/2);
```

$$(I = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, rotz = \begin{bmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, rotx = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & -0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}, roty = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 1.0 \\ -1.0 & 0.0 \end{bmatrix}$$

```
begin
latexify("I " ~ round4.(I(3))),
latexify("rotz" ~ round4.(mat3d[3]*I(3))),
latexify("rotx" ~ round4.(mat3d[1]*I(3))),
latexify("roty" ~ round4.(mat3d[2]*I(3)))
end
```

[3, 1, 2]

```
@bind rotind PlutoUI.Slider([[rand(1:3),rand(1:3),rand(1:3)] for i
=1:100],show_value=true,default=1)
```



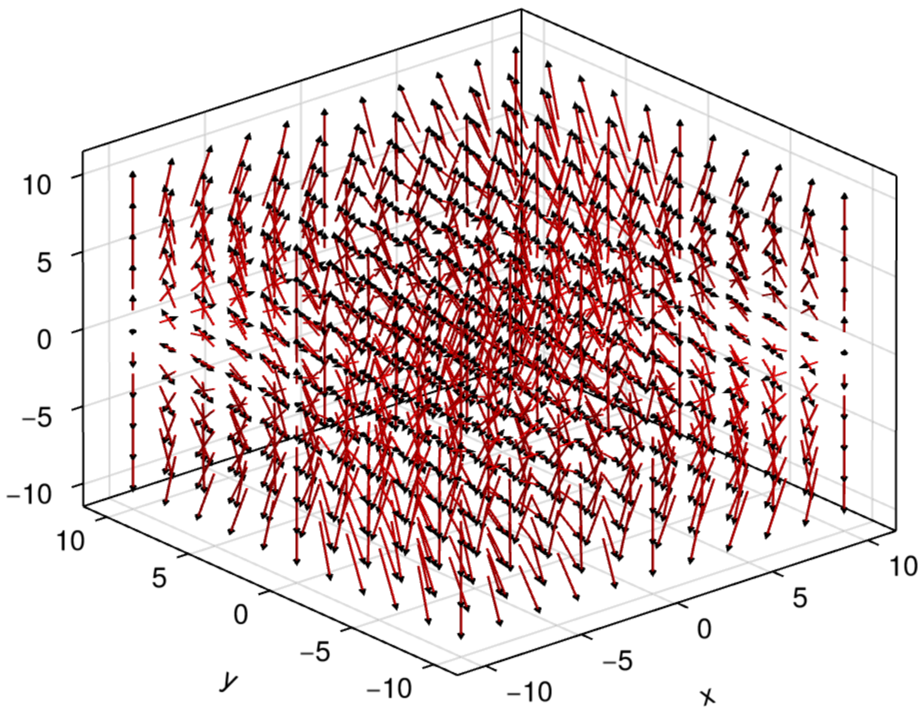

```
begin
ff2 = Figure(size=(500,500))
ax10=Axis3(ff2[1, 1], backgroundcolor = "beige")
vec0= [Point3f(x, y, z) for x in [0.0] for y in [0.0] for z in [0.0]]
vec10= map(x -> Vec3f(1.0, 0.5, 0.5), vec0)
v10=map(x -> Vec3f(mat3d[rotind[1]]*mat3d[rotind[2]]*mat3d[rotind[3]]*vec10[1]),vec0)
#vec10=Vec3f([10,10,10])
#strength10 = norm(v10)
#arrows!(ax10,vec0,vec10, fxaa=true, arrowsize =0.05 , lengthscale =1.0,linecolor =
:black, arrowcolor =:blue,linewidth = 0.025,align = :origin)

#arrows!(ax10,vec0,v10, fxaa=true, arrowsize =0.05 , lengthscale =1.0 ,linecolor =
:green, arrowcolor =:red,linewidth = 0.025,align = :origin)

arrows!(ax10,vec0,vec10,align=:origin,linewidth = 0.05,arrowsize = Vec3f(0.2, 0.2,
0.2))
arrows!(ax10,vec0,v10,align=:origin,linecolor=:red,linewidth = 0.05,arrowsize =
Vec3f(0.2, 0.2, 0.2))
xlims!(-2, 2)
ylims!(-2, 2)
zlims!(-2,2)
ff2
end
```

$$\left(orig = \begin{bmatrix} \begin{bmatrix} 1.0 \\ 0.5 \\ 0.5 \end{bmatrix} \end{bmatrix}, \ rotated = \begin{bmatrix} \begin{bmatrix} -0.5 \\ 0.5 \\ -1.0 \end{bmatrix} \end{bmatrix} \right)$$

```
begin
latexify("orig" ~ vec10),
latexify("rotated" ~ v10);
end
```




```
begin
ff3=Figure(size=(500,500))
ax11=Axis3(ff3[1, 1], backgroundcolor = "beige")
ps = [Point3f(x1, y1, z1) for x1 in -10:2:10 for y1 in -10:2:10 for z1 in -10:2:10];
L=10.0
ns = map(p -> Vec3f(sin(p[1]*π/L)*cos(p[2]*π/L), -sin(p[2]*π/L)*cos(p[1]*π/L),
p[3]*π/L), ps);
arrows!(
            ps, ns, fxaa=true, # turn on anti-aliasing
            linecolor = :red, arrowcolor = :black,
            linewidth = 0.1, arrowsize = Vec3f(0.3, 0.3, 0.4),
            align = :center)#,axis=(type=Axis3,))
ff3
end

```

MATRICES,NÚMEROS COMPLEJOS Y ROTACIONES

```
md"""
## MATRICES,NÚMEROS COMPLEJOS Y ROTACIONES
"""
```

$$(\delta = \delta(\beta,\gamma),\ \beta_1\gamma_1 - \beta_2\gamma_2 + (\beta_1\gamma_2 + \beta_2\gamma_1)i)$$

```
begin
@variables β[1:2], γ[1:2]
function δ(β,γ)
#   (β[1] +im*β[2])*(γ[1]+ im*γ[2])
    complex(β[1],β[2])*complex(γ[1],γ[2])
end
δ1=δ(β,γ)
#latexify("δ" ~ (β[1] +im*β[2])*(γ[1]+ im*γ[2])),
latexify(" δ = δ(β,γ) "), # ~ real(δ1) + (im)*imag(δ1))#+ im*δ1[2]])
#latexify(" = "),
latexify(δ(β,γ))

end
```

$$-1.0 + 0.0i$$

```
latexify(δ([0;1.0],[0.0,1.0]))
```

$$(\Gamma = \begin{bmatrix}\gamma_1 & -\gamma_2 \\ \gamma_2 & \gamma_1\end{bmatrix},\ \beta = \begin{bmatrix}\beta_1 & -\beta_2 \\ \beta_2 & \beta_1\end{bmatrix},\ \beta\Gamma = \begin{bmatrix}\beta_1\gamma_1 - \beta_2\gamma_2 & -\beta_1\gamma_2 - \beta_2\gamma_1 \\ \beta_1\gamma_2 + \beta_2\gamma_1 & \beta_1\gamma_1 - \beta_2\gamma_2\end{bmatrix})$$

```
begin
B(β)=[β[1] -β[2]; β[2] β[1] ]
Γ(γ)=[γ[1] -γ[2]; γ[2] γ[1] ]
latexify("Γ" ~ Γ(γ)),
latexify("β" ~ B(β)),
#latexify("βΓ" ~ B([0.0;1.0])*Γ([1.0,0.0]))
latexify("βΓ" ~ B(β)*Γ(γ))

end
```

```
[0.0, 1.0]
```

```
begin
    @variables ϕ
    function expΩ(ϕ)
        cos(ϕ)*I(2)+ sin(ϕ)*[0.0 -1.0;1.0 0.0]
    end
(B([0.0, 1.0]))*[1.0;0.0]
end
```

$$\begin{bmatrix} -1.0 & 0.0 \\ 0.0 & -1.0 \end{bmatrix}$$

```
latexify(round4.(expΩ(pi)))
#B([0.0, 1.0])*ϕ
```

```
mexp (generic function with 1 method)
```

```
begin
ϵ(ϕ)=(B([0.0, 1.0]))*ϕ
function mexp(ϵ)
    cos(ϕ)*I(2)+ sin(ϕ)*[0.0 -1.0;1.0 0.0]
end
end
```

$$\begin{bmatrix} 1.0 & e^{-\phi} \\ e^{\phi} & 1.0 \end{bmatrix}$$

```
exp.(ϵ(ϕ))
```

$$\begin{bmatrix} \cos\left(\phi \right) & - \sin\left(\phi \right) \\ \sin\left(\phi \right) & \cos\left(\phi \right) \end{bmatrix}$$

```
mexp(ϵ(ϕ))
```

```
0.789
```

```
round(rand(),sigdigits=3)
```

```
Enter cell code...
```

$$
\begin{aligned}
\frac{dx}{dt} &= y^2 - x \\
\frac{dy}{dt} &= \frac{x}{y} - y
\end{aligned}
$$

```
begin
lhs = ["dx/dt", "dy/dt"]
rhs = ["y^2 - x", "x/y - y"]
#display("text/latex", latexalign(lhs, rhs))
latexify(lhs,rhs)
end
#
```

FEOS!

```
md"""
# FEOS!
"""
```

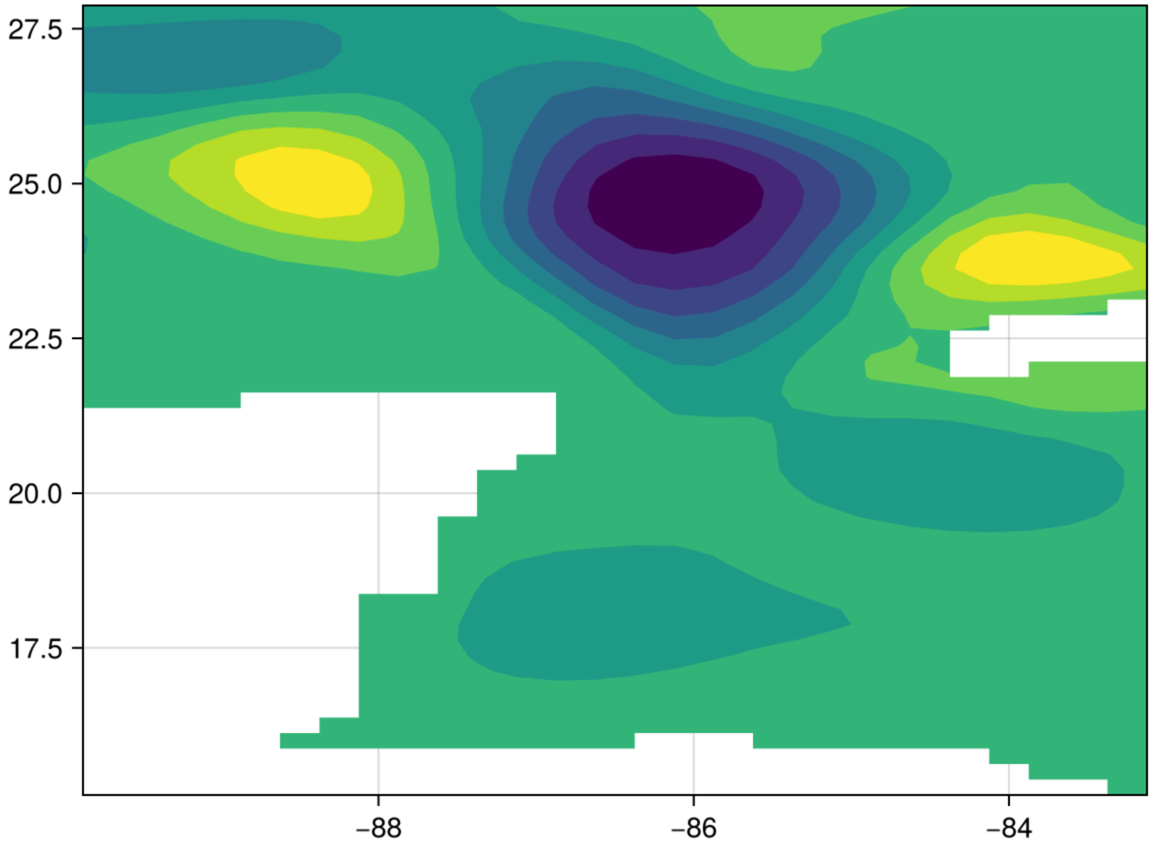
```
using Statistics
```

```
28×52 Matrix{Float64}:
 NaN           NaN           NaN           …   0.0144038    0.0165445    0.0155927
 NaN           NaN           NaN               0.0116998    0.0145214    0.0142532
 NaN           NaN           NaN               0.00450879   0.00782318   0.00916684
 NaN           NaN           NaN              -0.00601558  -0.00144783   0.00201051
 NaN           NaN           NaN              -0.0184822   -0.0120283   -0.00615532
 NaN           NaN           NaN           …  -0.0284349   -0.0216411   -0.013506
 NaN           NaN           NaN              -0.0319088   -0.0261673   -0.017726
   ⋮                                       ⋱                 ⋮
 NaN           NaN           NaN               0.00257328   0.00309879   0.00386748
 NaN           NaN            -0.00508         0.00240314   0.00334309   0.00424091
 NaN            -0.00640383   -0.00572849      0.00176435   0.0028995    0.00416571
 NaN            -0.00657268   -0.00528131  …   0.00135502   0.00262373   0.00396464
  -0.00746494   -0.00609906   -0.00492022      0.00152619   0.00306591   0.00426288
  -0.00589018   -0.00539526   -0.00437195      0.00248641   0.0041508    0.00509988
```

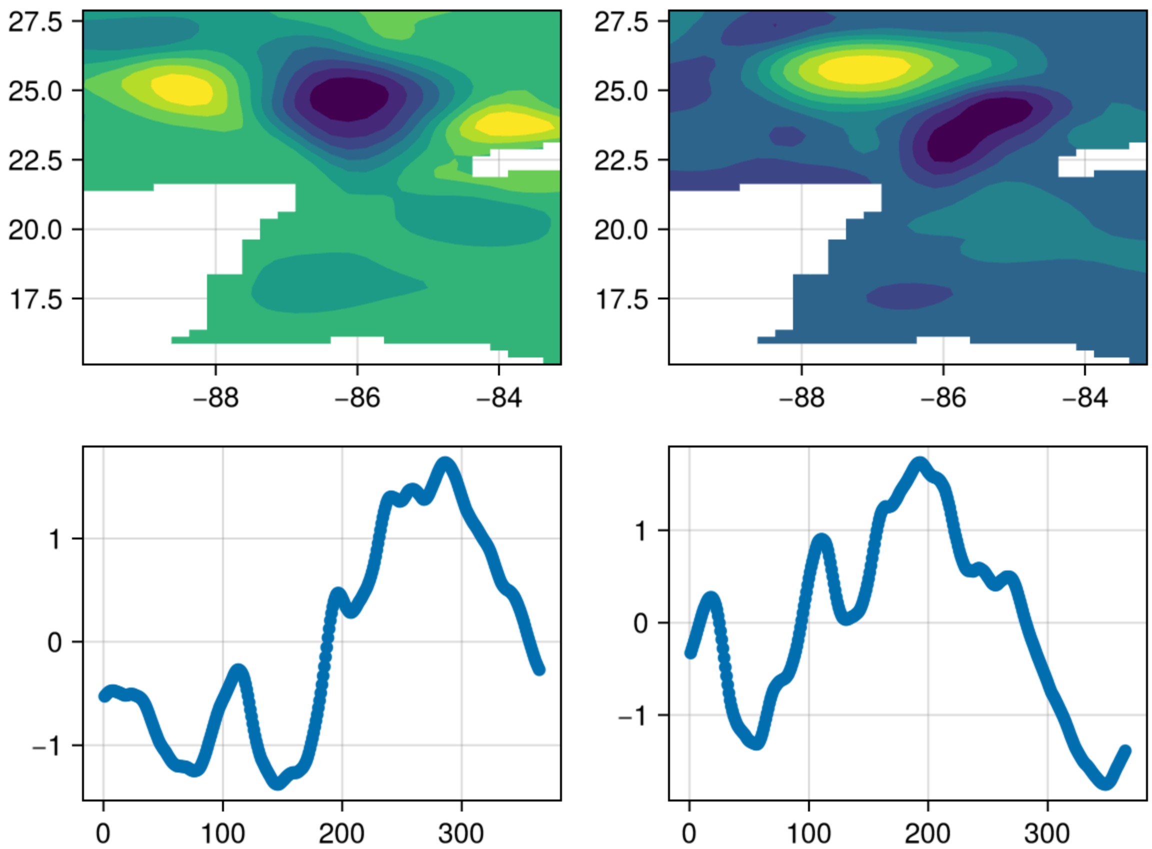
```julia
begin
ds = NCDataset("/Users/julios/JULIA/curso_datos_julia/tutorials/aviso_LC_2018.nc")
ds["adt"]
adt=reshape(ds["adt"],28*52,365)
# Find indices of NaN values
adt_nan_indices = findall(isnan, adt[:,1])

# Find indices of non-NaN values
adt_non_nan_indices = findall(!isnan, adt[:,1])

adt2=adt[adt_non_nan_indices,:]
size(adt2)
adt2=adt2';
size(adt2)
madt2=mean(adt2,dims=1);
δssh=adt2 .- madt2
ma,na=size(δssh)
mean(δssh,dims=1)
U,S,V=svd(δssh/sqrt(ma-1),full=false)
size(V)
scatter(S)
varianza=(S.*S)./sum(S.*S)
vareof1=round(varianza[1]*100)
vareof2=round(varianza[2]*100)
eof=zeros((length(adt[:,1]),4))*NaN
size(eof)
[eof[adt_non_nan_indices,i]=V[:,i] for i in 1:4]
size(eof)
eof[adt_nan_indices]
lon=ds["longitude"]
lat=ds["latitude"]
ssheof1=reshape(eof[:,1],28,52)
ssheof2=reshape(eof[:,2],28,52)
end
```




```
contourf(lon,lat,S[1].*ssheof1)
```




```
begin
fig = Figure()
#backgroundcolor = RGBf(0.98, 0.98, 0.98), size = (1000, 700))
axx1 = Axis(fig[1, 1])
axx2 = Axis(fig[1, 2])
axx3 = Axis(fig[2, 1])
axx4 = Axis(fig[2, 2])
contourf!(axx1,lon,lat,S[1].*ssheof1)#,fill=true,size=(400,600));
contourf!(axx2,lon,lat,S[2].*ssheof2)#,fill=true,size=(400,600));
plot!(axx3,U[:,1]*sqrt(ma-1),grid=true,label="PC1 = $vareof1%");
plot!(axx4,U[:,2]*sqrt(ma-1),grid=true,label="PC2 = $vareof2%");
fig
#grid_layout = vbox(
#    hbox(p1, p2),
#    hbox(p3, p4)
#)
#plot(p1,p2,p3,p4, layout = (2,2) )
#display(grid_layout)
#mean(U[:,1])
end
```