線性代數 + Julia + $\LaTeX$ 的學習筆記

Date: 2020-10-02

整個學習過程將以如下「線性代數」課程為主軸學習：

線性代數 台灣大學電機系 蘇柏青

本課程是線性代數的入門課程。線性代數係以「向量空間」(Vector Space)為核心概念之數學工具，擁有極廣泛之應用，非常值得理工商管等科系大學部同學深入修習，作為日後專業應用之基礎。

課程來源：**http://ocw.aca.ntu.edu.tw/ntu-ocw/index.php/ocw/cou/102S207**

學習目標

如下為幾個學習的子目標：

學科

- 線性代數 - 重新學習線性代數，了解重要概念的中文及英文詞彙及應用。

工具

- Julia - 深入學習，了解重要套件的應用及使用。
- Pluto - 隨之成長，作為撰寫學習記錄的工具。
- LaTeX - 隨緣學習，作為撰寫學習記錄的工具。
- Markdown - 隨緣學習，作為撰寫學習記錄的工具。

服務

- GitHub - 學習使用 GitHub 服務，並記錄學習歷程及分享學習內容。

單元 1．Basic Concepts on Matrices and Vectors

Matrix

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} = [a_{ij}] = M_{mn}$$

Matrix Addition

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 8 \end{bmatrix}$$

```
3×2 Array{Int64,2}:
 2  3
 4  5
 6  8
```

```
[1 2; 3 4; 5 6]+[1 1; 1 1; 1 2]
```

Scalar Multiplication

$$cA$$

$$3 \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

```
3×2 Array{Int64,2}:
  3   6
  9  12
 15  18
```

```
3 * [1 2; 3 4; 5 6]
```

```
3×2 Array{Int64,2}:
  3   6
  9  12
 15  18
```

```
3 .* [1 2; 3 4; 5 6]
```

Transpose

$$C = \begin{bmatrix} 7 & 9 \\ 18 & 31 \\ 52 & 68 \end{bmatrix} \Rightarrow C^T = \begin{bmatrix} 7 & 18 & 52 \\ 9 & 31 & 68 \end{bmatrix}$$

```
2×3 LinearAlgebra.Adjoint{Int64,Array{Int64,2}}:
 7  18  52
 9  31  68
```

```
let
    C=[7 9; 18 31; 52 68]
    C'
end
```

Vectors

Row Vector:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

Column Vector:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}^T$$

The ith componet of $\mathbf{v}$

$$v_i$$

```
1×4 Array{Int64,2}:
 1  2  3  4
```

```
[ 1 2 3 4]
```

```
▶Int64[1,  2,  3,  4]
```

```
[1; 2; 3; 4;]
```

```
4×1 LinearAlgebra.Adjoint{Int64,Array{Int64,2}}:
 1
 2
 3
 4
```

```
[ 1 2 3 4]'
```

Linear Combination

A $\textcolor{red}{linear\ combination}\ of\ vectors$ $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ is a vector of the form

$$c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_k\mathbf{u}_k$$

where $c_1, c_2, \ldots, c_k$ are scalars. These scalars are called the $\textcolor{red}{coefficients}$ of the linear combination.

Standard Vectors

The standard vectors of R^n are defined as

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, e_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Matrix-Vector Product

$$Av = v_1 a_1 + v_2 a_2 + \cdots + v_n a_n$$

```
Int64[23,  53,  83]
```

```
let
    A=[1 2; 3 4; 5 6]
    v=[7;8]
    A*v
end
```

Identity Matrix

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Stochastic Matrix

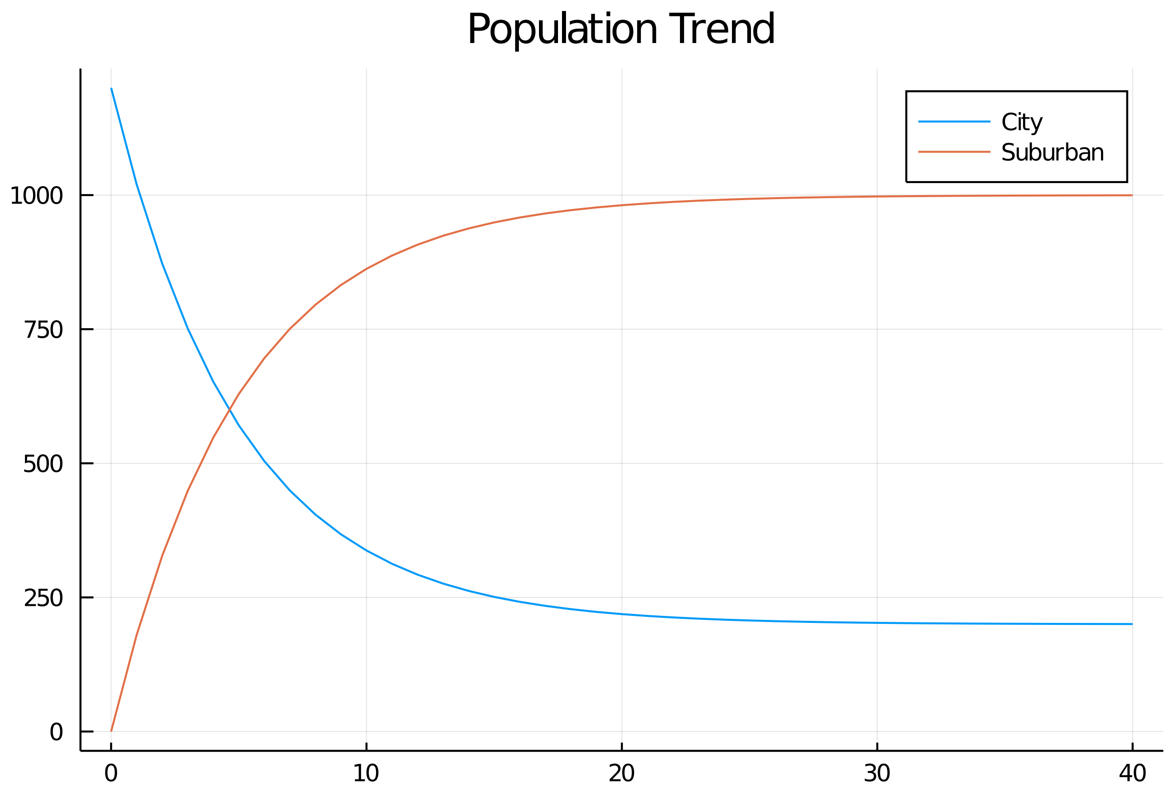
$$A = \begin{bmatrix} 0.85 & 0.03 \\ 0.15 & 0.97 \end{bmatrix}$$

Slide to set number of **years**: 40

Slide to set population of **city**: 1200

Slide to set population of **suburban**: 0

```
begin
    u01xslider = @bind u01x Slider(1:100; default=40, show_value=true)
    u01cslider = @bind u01c Slider(0:1200; default=1200, show_value=true)
    u01sslider = @bind u01s Slider(0:1200; default=0, show_value=true)
    md"""
    Slide to set number of **years**: $(u01xslider)

    Slide to set population of **city**: $(u01cslider)

    Slide to set population of **suburban**: $(u01sslider)
    """
end
```




```
let
    x=u01x # Number of Years (x)
    pc=u01c # Population of City
    ps=u01s # Population of Suburban
    A=[0.85 0.03; 0.15 0.97]
    #=
    # p0 Population in year 0
    p0=[500; 700]
    p1=A*p0
    p2=A*(p1)
    p3=A*(p2)
    p4=A*(p3)
    p5=A*(p4)
    x=0:5
    Y=hcat(p0, p1, p2, p3, p4, p5)
    plot(x, Y', title = "Population", label = ["City" "Suburban"])
    =#
    p=[pc; ps]
    Y=p
    for i in 1:x
        p=A*p
        Y=hcat(Y, p)
    end
    plot(0:x, Y', title = "Population Trend", label = ["City" "Suburban"])
end
```

單元 2 · System of Linear Equations

System of Linear Equations

$$A = \begin{bmatrix} 1 & -2 & -1 \\ 3 & -6 & -5 \\ 2 & -1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix}$$

$$Ax = b$$

Solves Ax = b by (essentially) Gaussian elimination (Julia \ Operator):

$$x = A \setminus b$$

```
[-4.000000000000001, -5.000000000000001, 3.000000000000001]
[3.0, 3.0, 0.0]
true
```

```julia
# Solve System of Linear Equations
let
    with_terminal() do
        A=[1 -2 -1; 3 -6 -5; 2 -1 1]
        b=[3; 3; 0]
        x=A \ b
        println(x)
        println(A*x)
        println(A*x == b)
    end
end
```

Row Echelon Form & Reduced Row Echelon Form

```
[0.4037433155080213, -1.2112299465240637, 0.11229946524064169, 1.4812834224598934, 2.0]
[6.999999999999998, 9.000000000000002, 2.0, 0.0]
false
```

```julia
# Solve System of Linear Equations
let
    with_terminal() do
        A=[1 -3 0 2 0; 0 0 1 6 0; 0 0 0 0 1; 0 0 0 0 0]
        b=[7; 9; 2; 0]
        x=A \ b
        println(x)
        println(A*x)
        println(A*x == b)
    end
end
```

單元 3．Gaussian Elimination

實作參考：

Gaussian-elimination.pdf

Numerical Analysis by Julia Series 1 — Gauss Elimination | by Treee July | Medium

對列及行的參照：

```
3×3 Array{Int64,2}:
 1  2  3
 4  5  6
 7  8  9
```

```julia
let
    A=[ 1 2 3; 4 5 6; 7 8 9]
end
```

```
Array{Any}((7,))
  1: String "A: [1 2 3; 4 5 6; 7 8 9]"
  2: String "A[1, 1]: 1"
  3: String "A[end, end]: 9"
  4: String "r1: [1, 2, 3]"
  5: String "∑Ai: [12, 15, 18]"
  6: String "c1: [1, 4, 7]"
  7: String "∑Aj: [6, 15, 24]"
```

```julia
let
    o=[]
    # Matrix
    A=[ 1 2 3; 4 5 6; 7 8 9]
    push!(o, @sprintf("A: %s", A))
    # Elements
    push!(o, @sprintf("A[1, 1]: %s", A[1, 1]))
    push!(o, @sprintf("A[end, end]: %s", A[end, end]))
    # Row
    r1=A[1,:]
    push!(o, @sprintf("r1: %s", r1))
    ∑Ai=A[1,:]+A[2,:]+A[3,:]
    push!(o, @sprintf("∑Ai: %s", ∑Ai))
    # Column
    c1=A[:,1]
    push!(o, @sprintf("c1: %s", c1))
    ∑Aj=A[:,1]+A[:,2]+A[:,3]
    push!(o, @sprintf("∑Aj: %s", ∑Aj))
    with_terminal(dump, o)
end
```

```
👍 選定之輸出方案： 1) 容易以 do ... end 區塊包裝 3) 轉貼程式碼到他處不用修改

2020年10月 2日 週五 17時24分43秒 CST
A: Array{Int64}((3, 3)) [1 2 3; 4 5 6; 7 8 9]
A[1, 1]: Int64 1
A[end, end]: Int64 9
r1: Array{Int64}((3,)) [1, 2, 3]
∑Ai: Array{Int64}((3,)) [12, 15, 18]
c1: Array{Int64}((3,)) [1, 4, 7]
∑Aj: Array{Int64}((3,)) [6, 15, 24]

      10月 2020
日 一 二 三 四 五 六
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

```
let
    with_terminal() do
        println("👍 選定之輸出方案： 1) 容易以 do ... end 區塊包裝 3) 轉貼程式碼到他處不用修改\n")
        # Get Current Time
        command=`date`
        run(command)
        # Matrix
        A=[ 1 2 3; 4 5 6; 7 8 9]
        print("A: "); dump(A)
        # Elements
        print("A[1, 1]: "); dump(A[1, 1])
        print("A[end, end]: "); dump(A[end, end])
        # Row
        r1=A[1,:]
        print("r1: ");  dump(r1)
        ∑Ai=A[1,:]+A[2,:]+A[3,:]
        print("∑Ai: "); dump(∑Ai)
        # Column
        c1=A[:,1]
        print("c1: ");  dump(c1)
        ∑Aj=A[:,1]+A[:,2]+A[:,3]
        print("∑Aj: "); dump(∑Aj)
        println()
        run(`cal -h`)
    end
end
```

```
A: Array{Int64}((3, 3)) [1 2 3; 4 5 6; 7 8 9]
A[1, 1]: Int64 1
A[end, end]: Int64 9
r1: Array{Int64}((3,)) [1, 2, 3]
∑Ai: Array{Int64}((3,)) [12, 15, 18]
c1: Array{Int64}((3,)) [1, 4, 7]
∑Aj: Array{Int64}((3,)) [6, 15, 24]
```

```
let
    Text() do io
        # Matrix
        A=[ 1 2 3; 4 5 6; 7 8 9]
        print(io, "A: "); dump(io, A)
        # Elements
        print(io, "A[1, 1]: "); dump(io, A[1, 1])
        print(io, "A[end, end]: "); dump(io, A[end, end])
        # Row
        r1=A[1,:]
        print(io, "r1: ");  dump(io, r1)
        ∑Ai=A[1,:]+A[2,:]+A[3,:]
        print(io, "∑Ai: "); dump(io, ∑Ai)
        # Column
        c1=A[:,1]
        print(io, "c1: ");  dump(io, c1)
        ∑Aj=A[:,1]+A[:,2]+A[:,3]
        print(io, "∑Aj: "); dump(io, ∑Aj)
    end
end
```

A: [1 2 3; 4 5 6; 7 8 9]

A[1, 1]: 1

A[end, end]: 9

r1: [1, 2, 3]

∑Ai: [12, 15, 18]

c1: [1, 4, 7]

∑Aj: [6, 15, 24]

```julia
let
		# Matrix
		A=[ 1 2 3; 4 5 6; 7 8 9]
		# Row
		r1=A[1,:]
		∑Ai=A[1,:]+A[2,:]+A[3,:]
		# Column
		c1=A[:,1]
		∑Aj=A[:,1]+A[:,2]+A[:,3]
	md"""
	A: $(Text(A))

	A[1, 1]: $(Text(A[1, 1]))

	A[end, end]: $(Text(A[end, end]))

	r1: $(Text(r1))

	∑Ai: $(Text(∑Ai))

	c1: $(Text(c1))

	∑Aj: $(Text(∑Aj))
	"""
end
```

單元 4 · The language of set theory

Subset

$$Let\ S_1 = \{a, b, c, d, e\},\ S_2 = a, b, e$$

$$S_2 \subset S_1\ means$$

$$\forall x \in S_2,\ x\ is\ also \in S_1.$$

```
🖐 Give Me Five, 原來集合在 Julia 裏頭是長這樣子喔！☯
Set{String}
  dict: Dict{String,Nothing}
    slots: Array{UInt8}((16,)) UInt8[0x00, 0x00, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x0
1, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01]
    keys: Array{String}((16,))
      1: #undef
      2: #undef
      3: #undef
      4: String "c"
      5: String "e"
      ...
      12: #undef
```

```
        13: #undef
        14: String "a"
        15: #undef
        16: String "d"
      vals: Array{Nothing}((16,))
        1: Nothing nothing
        2: Nothing nothing
        3: Nothing nothing
        4: Nothing nothing
        5: Nothing nothing
        ...
        12: Nothing nothing
        13: Nothing nothing
        14: Nothing nothing
        15: Nothing nothing
        16: Nothing nothing
      ndel: Int64 0
      count: Int64 5
      age: UInt64 0x0000000000000005
      idxfloor: Int64 1
      maxprobe: Int64 0
  Set(["c", "e", "b", "a", "d"])
  Set(["e", "b", "a"])
  true
  Set(["c", "e", "b", "a", "d"])
  Set(["e", "b", "a"])
  Set(["c", "d"])
  Set{String}()
```

```
let
    with_terminal() do
        s1=Set(["a", "b", "c", "d", "e"])
        println("🖐 Give Me Five, 原來集合在 Julia 裏頭是長這樣子喔！☯")
        dump(s1)
        println(s1)
        s2=Set(["a", "b", "e"])
        println(s2)
        # subset
        println(⊆(s2, s1))
        # union set
        println(∪(s1, s2))
        # intersection set
        println(∩(s1, s2))
        # difference set
        println(setdiff(s1, s2))
        println(setdiff(s2, s1))
    end
end
```

單元 5 · Span of a Set of Vectors

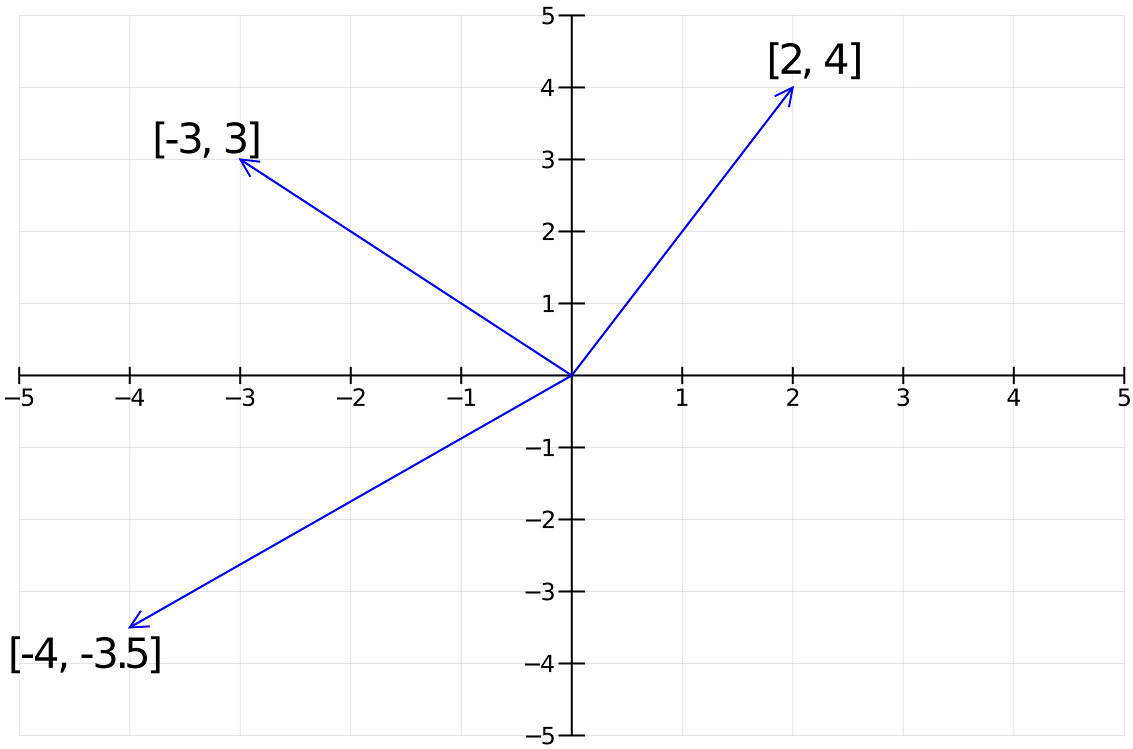
```
md"""
## 單元 5 · Span of a Set of Vectors
"""
```

實作參考：

Linear Algebra – Quantitative Economics with Julia

```
md"""
```

```
實作參考：

[Linear Algebra — Quantitative Economics with Julia]
(https://julia.quantecon.org/tools_and_techniques/linear_algebra.html)
"""
```

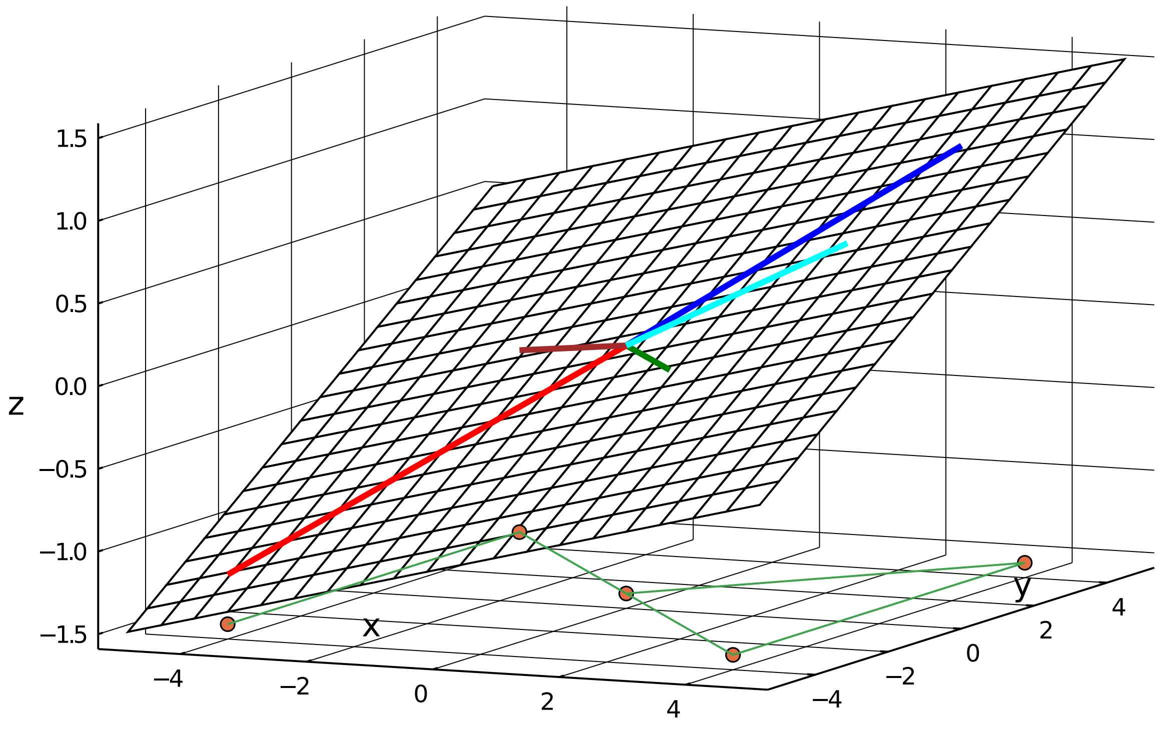



```
let
    x_vals = [0 0 0 ; 2 -3 -4]
    y_vals = [0 0 0 ; 4 3 -3.5]

    plot(x_vals, y_vals, arrow = true, color = :blue,
         legend = :none, xlims = (-5, 5), ylims = (-5, 5),
         annotations = [(2.2, 4.4, "[2, 4]"),
                        (-3.3, 3.3, "[-3, 3]"),
                        (-4.4, -3.85, "[-4, -3.5]")],
         xticks = -5:1:5, yticks = -5:1:5,
         framestyle = :origin)
end
```

Slide to set **i**: 1

Slide to set **j**: 5

```
begin
    u05islider = @bind u05i Slider(1:5; default=1, show_value=true)
    u05jslider = @bind u05j Slider(1:5; default=5, show_value=true)
    md"""
    Slide to set **i**: $(u05islider)

    Slide to set **j**: $(u05jslider)
    """
end
```




```
let
    i=u05i
    j=u05j
    # fixed linear function, to generate a plane
    f(x, y) = 0.2x + 0.1y

    # lines to vectors
    x_vec = [0 0 0 0 0; 3 3 -4 -4 3.5]
    y_vec = [0 0 0 0 0; 4 -4 -4 4 0]
    z_vec = [0 0 0 0 0; f(3, 4) f(3, -4) f(-4,-4) f(-4, 4) f(3.5, 0)]
    color = [:blue :green :red :brown :cyan]

    # draw the plane
    n = 20
    grid = range(-5, 5, length = n)
    z2 = [ f(grid[row], grid[col]) for row in 1:n, col in 1:n ]
    # wireframe(grid, grid, z2, fill = :blues, gridalpha =1 )
    plot(grid, grid, z2, fill = :blues, gridalpha = 1, lindwidth = 0.5, seriestype =
:wireframe)
    # plot(grid, grid, z2, fill = :blues, gridalpha = 1, lindwidth = 0.5, seriestype =
:surface)
    # Dots
    # plot!([0; 4; 4; -4; -4], [0; 4; -4; 4; -4], [-1.5; -1.5; -1.5; -1.5; -1.5], labels =
"", seriestype = :scatter3d)
    p = [ 0 0 -1.5; 4 4 -1.5; 4 -4 -1.5; -4 4 -1.5; -4 -4 -1.5]' # Transpose
    plot!(p[1, i:j], p[2, i:j], p[3, i:j], labels = "", seriestype = :scatter3d)
    plot!(p[1, i:j], p[2, i:j], p[3, i:j], labels = "", seriestype = :path3d)
    # Vectors
    plot!(x_vec[:, i:j], y_vec[:, i:j], z_vec[:, i:j], color = color[:,i:j], linewidth = 3,
xlabel = "x", ylabel = "y", zlabel = "z", labels = "", colorbar = false)
    # plot!(x_vec, y_vec, z_vec, color = color[:,i:j], linewidth = 3, xlabel = "x", ylabel =
"y", zlabel = "z", labels = "", colorbar = false)
end
```

單元 6．Linear Dependence and Linear Independence

```
[-0.09523809523809523, -0.1295238095238095, -0.08761904761904767]
[-0.2666666666666666, 0.6666666666666671, -1.2000000000000006, -0.6666666666666671]
false
```

```
let
    with_terminal() do
        A=[1 2 -1; -1 1 -8; 2 -1 13; 1 -1 8]
        b=[0; 1; -2; 1]
        x=A \ b
        println(x)
        println(A*x)
        println(A*x == b)
    end
end
```

```
[0.0, 0.0, 0.0, -0.0, 0.0]
[0.0, 0.0]
true
```

```
let
    with_terminal() do
        A=[1 -4 2 -1 2; 2 -8 3 2 1]
        b=[0; 0]
        x=A \b
        println(x)
        println(A*x)
        println(A*x == b)
    end
end
```

單元 7 · Matrix Multiplication

Matrix Multiplication

$$Let\ v, x, y \in R^n. Suppose\ A\ and\ B\ are\ n \times n\ matrices.$$

$$x = Bv$$

$$y = Ax$$

$$\Downarrow$$

$$y = Cv = A(Bv) = (AB)v$$

```
Array{Int64}((2, 2)) [3 -1; 2 0]
Array{Int64}((2, 2)) [2 3; 0 1]
```

```
let
    with_terminal() do
        A=[1 2; 1 1]
        B = [ 1 1; 1 -1]
```

```
·         dump(A * B)
·         dump(B * A)
·     end
· end
```

單元 8 · Invertibility and Elmentary Matrices

Inverse

An n × n matrix A is called invertible if there exists an n × n matrix B such that $AB = BA = I_n$. In this case, B is called an inverse of A.

$$\begin{array}{rl} A = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} & \\ & \Rightarrow \\ B = \begin{bmatrix} -5 & 2 \\ 3 & -1 \end{bmatrix} & \end{array} \quad \begin{array}{l} AB = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} -5 & 2 \\ 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2 \\ BA = \begin{bmatrix} -5 & 2 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2 \end{array}$$

```
Array{Int64}((2, 2)) [1 2; 3 5]
Array{Float64}((2, 2)) [-4.999999999999997 1.9999999999999999; 2.9999999999999987 -0.9999
999999999996]
Array{Float64}((2, 2)) [-5.0 2.0; 3.0 -1.0]
Float64 1.0
Float64 0.0
Float64 -0.0
Float64 1.0
Float64 1.0
Float64 0.0
Float64 0.0
Float64 1.0
```

```
· let
·     with_terminal() do
·         A=[1 2; 3 5]
·         B=inv(A)
·         dump(A)
·         dump(B)
·         dump(round.(B))
·         dump.(round.(A*B))
·         dump.(round.(B*A))
·     end
· end
```

疑：為什麼只是近似的值？因為 3?

單元 9 · Column Correspondence Theorem

實作參考：

[blegat/RowEchelon.jl: Small package containing the rref fonction for computing the reduced row echelon form of the matrix A]()

可做為 module 及 RREF 計算實作參考。 提供的 function 為 rref, rref! rref*with*pivots, rref*with*pivots!,

Reduced Row Echelon Form (RREF)

```
Using RREF:
Array{Int64}((4, 6)) [1 2 … 1 2; -1 -2 … 3 6; 2 4 … 0 3; -3 -6 … 3 9]
[1 2 -1 2 1 2; -1 -2 1 2 3 6; 2 4 -3 2 0 3; -3 -6 2 0 3 9]
Array{Float64}((4, 6)) [1.0 2.0 … -1.0 -5.0; 0.0 0.0 … 0.0 -3.0; 0.0 0.0 … 1.0 2.0; 0.0 0.0 … 0.0 0.0]
[1.0 2.0 0.0 0.0 -1.0 -5.0; 0.0 0.0 1.0 0.0 0.0 -3.0; 0.0 0.0 0.0 1.0 1.0 2.0; 0.0 0.0 0.0 0.0 0.0 0.0]
Using \:
Array{Float64}((5,)) [-1.0, -1.0, -3.0, 1.0, 1.0]
[-1.0, -1.0, -3.0, 1.0, 1.0]
[2.0, 6.0, 3.0, 9.0]
Using \ with RREF:
Array{Float64}((5,)) [-1.0, -1.0, -3.0, 1.0, 1.0]
[-1.0, -1.0, -3.0, 1.0, 1.0]
[-5.0, -3.0, 2.0, 0.0]
```

```
let
    with_terminal() do
        println("Using RREF:")
        A=[ 1 2 -1 2 1 2; -1 -2 1 2 3 6; 2 4 -3 2 0 3; -3 -6 2 0 3 9]
        dump(A)
        println(Text(A))
        B=rref(A)
        dump(round.(B))
        println(Text(round.(B)))

        println("Using \\:")
        A1=A[:, 1:5]
        b1=A[:, 6]
        x=A1 \ b1
        dump(round.(x))
        println(Text(round.(x)))
        println(Text(round.(A1*x)))

        println("Using \\ with RREF:")
        A2=B[:, 1:5]
        b2=B[:, 6]
        y=A2 \ b2
        dump(round.(y))
        println(Text(round.(y)))
        println(Text(round.(A2*y)))
    end
end
```

單元 10 · The Inverse of a Matrix

Matrix Inversion

$$[A \quad I_3] = \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 5 & 6 & 0 & 1 & 0 \\ 3 & 4 & 8 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -16 & 4 & 3 \\ 0 & 1 & 0 & -2 & 1 & 0 \\ 0 & 0 & 1 & 7 & -2 & -1 \end{array}\right] = [I_3 \quad B]$$

```
[1 2 3 1 0 0; 2 5 6 0 1 0; 3 4 8 0 0 1]
[1.0 0.0 0.0 -16.0 4.0 3.0; 0.0 1.0 0.0 -2.0 1.0 -0.0; 0.0 0.0 1.0 7.0 -2.0 -1.0]
[-16.0 4.0 3.0; -2.0 1.0 -0.0; 7.0 -2.0 -1.0]
[-16.0 4.0 3.0; -2.0 1.0 0.0; 7.0 -2.0 -1.0]
```

```
let
    with_terminal() do
        AI=[ 1 2 3 1 0 0; 2 5 6 0 1 0; 3 4 8 0 0 1]
        println(AI)
        IB=round.(rref(AI))
        println(IB)
        C=IB[:, 4:6]
        println(C)
        D=round.(inv(AI[:, 1:3]))
        println(D)
    end
end
```

<<<

```
md"""
### <<<
"""
```

參考資料

Linear Algebra

[ ] 線性代數 - 臺大開放式課程 (NTU OpenCourseWare)

[ ] Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares

> Julia language companion

Julia

[ ] Introduction to Julia

> [ ] Advanced topics

[ ] Julia for Data Science

[ ] 18.S191 Introduction to Computational Thinking

Linear Algebra – Quantitative Economics with Julia

> QuantEcon.cheatsheet/julia-cheatsheet.pdf

cheatsheets/plotsjl-cheatsheet.pdf

Unicode Input · The Julia Language

Visualizing Graphs in Julia using Plots and PlotRecipes – Tom Breloff

Pluto

fonsp/Pluto.jl: 🎈 Simple reactive notebooks for Julia

Docstrings · PlutoUI.jl

$\LaTeX$

LaTeX syntax · Documenter.jl

Documentation - Overleaf, Online LaTeX Editor

LaTeX - Mathematical Python

LaTeX help 1.1 - Table of Contents

List of mathematical symbols - Wikiwand

Markdown

Markdown Cheatsheet · adam-p/markdown-here Wiki

Markdown · The Julia Language

GitHub

[ ] Hello World · GitHub Guides

其他

三度辭典網 > 術語中英雙語詞典

```
md"""
## 參考資料
### Linear Algebra

[ ] [線性代數 - 臺大開放式課程 (NTU OpenCourseWare)]
(http://ocw.aca.ntu.edu.tw/ntu-ocw/index.php/ocw/cou/102S207/3)

[ ] [Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares]
(http://vmls-book.stanford.edu/)
> [Julia language companion](http://vmls-book.stanford.edu/vmls-julia-companion.pdf)

### Julia

[ ] [Introduction to Julia]
(https://juliaacademy.com/courses/enrolled/375479)
```

```

> [ ] Advanced topics

[ ] [Julia for Data Science]
(https://juliaacademy.com/courses/enrolled/937702)

[ ] [18.S191 Introduction to Computational Thinking]
(https://computationalthinking.mit.edu/Fall20/)

[Linear Algebra — Quantitative Economics with Julia]
(https://julia.quantecon.org/tools_and_techniques/linear_algebra.html)

> [QuantEcon.cheatsheet/julia-cheatsheet.pdf]
(https://github.com/QuantEcon/QuantEcon.cheatsheet/blob/master/julia/julia-cheatsheet.pdf)

[cheatsheets/plotsjl-cheatsheet.pdf]
(https://github.com/sswatson/cheatsheets/blob/master/plotsjl-cheatsheet.pdf)

[Unicode Input · The Julia Language]
(https://docs.julialang.org/en/v1/manual/unicode-input/)

[Visualizing Graphs in Julia using Plots and PlotRecipes — Tom Breloff]
(http://www.breloff.com/Graphs/)

### Pluto
[fonsp/Pluto.jl: 🎈 Simple reactive notebooks for Julia]
(https://github.com/fonsp/Pluto.jl)



[Docstrings · PlutoUI.jl]
(https://juliahub.com/docs/PlutoUI/abXFp/0.6.3/autodocs/)

### $$\LaTeX$$

[LaTeX syntax · Documenter.jl]
(https://juliadocs.github.io/Documenter.jl/v0.7/man/latex.html)

[Documentation - Overleaf, Online LaTeX Editor]
(https://www.overleaf.com/learn/latex/Main_Page)

[LaTeX - Mathematical Python]
(https://www.math.ubc.ca/~pwalls/math-python/jupyter/latex/)

[LaTeX help 1.1 - Table of Contents]
(http://www.emerson.emory.edu/services/latex/latex_toc.html)

[List of mathematical symbols - Wikiwand]
(https://www.wikiwand.com/en/List_of_mathematical_symbols)

### Markdown
[Markdown Cheatsheet · adam-p/markdown-here Wiki]
(https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)

[Markdown · The Julia Language]
(https://docs.julialang.org/en/v1/stdlib/Markdown/)

### GitHub

[ ] [Hello World · GitHub Guides]
(https://guides.github.com/activities/hello-world/)

### 其他
[三度辭典網 > 術語中英雙語詞典]
(https://www.3du.tw/term/)
```

- """