Rapport de laboratoire 1: méthode de Gram-Schmidt

MTH8211

Amami Yasmine

```
using Pkg
Pkg.activate("rapport_env")    # activate a virtual environment
using LinearAlgebra
```

Implémentation de Gram-Schmidt classique

```
function gram_schmidt(us)
    vs = copy(us)
    for i = 1 : length(vs)
        v = vs[i]
        for j = 1 : i - 1
           v .-= dot(us[i], vs[j]) * vs[j]
        end
        v .= v / norm(v)
    end
    return vs
end
```

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

Illustration de l'échec en cas de dépendance linéaire

```
vector = [0.0 1.0 1.0; 1.0 1.0 2.0; 0.0 0.0 0.0]
println("Exemple de matrice qui presente une dependance lineaire :")
display(vector)
vector_cols = [vector[:, j] for j in 1:size(vector, 2)]
result = gram_schmidt(vector_cols) # on va voir l'erreur ici
```

```
Exemple de matrice qui presente une dependance lineaire :
```

```
3×3 Matrix{Float64}:
 0.0  1.0  1.0
 1.0  1.0  2.0
 0.0  0.0  0.0
```

```
3-element Vector{Vector{Float64}}:
 [0.0, 1.0, 0.0]
 [1.0, 0.0, 0.0]
 [NaN, NaN, NaN]
```

Explication : la troisieme colonne est dependante des deux premieres colonnes. Une fois que la projection est realise le troisieme vecteur devient pratiquement nul. Or la division par une valeur proche de zero ou zero lors de la normalisation cause une erreur.

Implémentation de Gram-Schmidt plus robuste

```julia
function gram_schmidt_robuste(us; tol=1e-10)
    vs = copy(us)
    independents = Int[]
    for i = 1:length(vs)
        v = vs[i]
        for j in independents
            v .-= dot(us[i], vs[j]) * vs[j]
        end
        if norm(v) > tol
            v .= v / norm(v)
            push!(independents, i)
        else
            v .= zeros(eltype(v), length(v))
        end
    end
    return vs, independents
end
```

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

Certains Avantages : - identification explicite des vecteurs lineairement dependants - pas d'erreurs liees a la division par zero

Certains Inconvenients : - besoin de determiner le bon seuil de tolerance - sensibilite et possible erreur de la methode pour les vecteurs qui sont presque dependants

Démonstration de la nouvelle implémentation

```
using Test

vectors = [[1.0, 0.0, 0.0], [0.0, 1.0, 0.0], [2.0, 3.0, 0.0]]
orthogonal, independent = gram_schmidt_robuste(vectors)

@test length(independent) == 2
@test 1 in independent && 2 in independent
@test !(3 in independent)
@test norm(orthogonal[3]) == 0
@test isapprox(dot(orthogonal[1], orthogonal[2]), 0, atol=1e-10)
```

```
Test Passed
```