TP Maple 5 | Espaces préhilbertiens. Décomposition QR.

Le logiciel dispose des outils classiques de calcul vectoriel euclidien (et hermitien) dans $\mathbb{R}^n$ (resp. $\mathbb{C}^n$). Dans le cas d'un produit scalaire sur un espace E de dimension finie n, on pourra définir le produit scalaire par une application $\psi : E^2 \to \mathbb{R}$. On exposera à la fin de ce TP les méthodes de Schmidt et d'Householder de décomposition QR d'une matrice réelle.

On utilisera la bibliothèque **LinearAlgebra** comportant les commandes suivantes d'algèbre linéaire et bilinéaire :

```
> with(LinearAlgebra);
```

Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, LA|_Main, LUDecomposition, LeastSquares, LinearSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip

1. Calculs dans l'espace euclidien canonique $\mathbb{R}^n$

Nous n'avons retenu ici que les outils euclidiens les plus importants : le produit scalaire, la norme euclidienne, le produit vectoriel (en dimension trois) et le procédé d'orthogonalisation (d'une famille libre) de Gram-Schmidt.

```
> u:=<1,2,3>: v:=<-1,-1,1>: DotProduct(u,v), CrossProduct(u,v), Norm(u,2);
```

$$0,\ \begin{bmatrix}5\\-4\\1\end{bmatrix},\ \sqrt{14}$$

```
> u1:=<1,1,1>: u2:=<1,1,0>: u3:=<0,1,1>: GramSchmidt([u1,u2,u3]);
```

$$\left[\begin{bmatrix}1\\1\\1\end{bmatrix},\begin{bmatrix}1/3\\1/3\\-2/3\end{bmatrix},\begin{bmatrix}-1/2\\1/2\\0\end{bmatrix}\right]$$

Structure euclidienne canonique de $\mathbb{R}^n$

- **DotProduct** : produit scalaire ;
- **Norm(u,2)** : norme euclidienne de u ;
- **CrossProduct** : produit vectoriel (en dimension trois) ;
- **GramSchmidt([v_1,v_2,...,v_n])** : b.o. obtenue à partir de $(v_1,\dots,v_n)$ par le procédé de Gram-Schmidt ;

2. Calculs dans un espace préhilbertien quelconque

Dans le cadre d'un espace euclidien quelconque E, on commencera par définir sous la forme d'une fonction $\psi : E^2 \to \mathbb{R}$ le produit scalaire de E. Il sera parfois utile de créer les procédures d'orthonormalisation de Gram-Schmidt [1]...

1. Attention, le nom **GramSchmidt** est protégé sous Maple.

```
> MyGramSchmidt:=proc(BASE)
  local BON,k,v,w,m;
  BON:=[BASE[1]/sqrt(psi(BASE[1], BASE[1]))]:
  for k from 2 to nops(BASE)
                    do
                       m:=k-1: v:=sum(psi(BASE[k],BON[p])*BON[p],p=1..m):
                       w:=(BASE[k]-v)/sqrt(psi(BASE[k]-v,BASE[k]-v)):
                       BON:=[op(BON),w]:
                    od:
  BON:
  end:
```

...et de projection orthogonale sur un s.e.v. dont on connaît une base (pas nécessairement orthonormée).

```
> ProjectionOrthogonaleVersion1:=proc(u,BASE)
  local BON;
  BON:=MyGramSchmidt(BASE):
  sum(psi(u,BON[p])*BON[p],p=1..nops(BASE)):
  end:
```

On pourra également calculer la projection orthogonale $\pi_F(u)$ de u sur $F = \text{vect}(f_1, \ldots, f_m)$ sans passer par une orthonormalisation de Gram-Schmidt en résolvant le système d'inconnue $v \in F$ suivant :

$$\forall 1 \leqslant k \leqslant m,\ \psi(u - v, f_k) = 0.$$

Son unique solution est $\pi_F(u)$.

```
> ProjectionOrthogonaleVersion2:=proc(u,BASE)
  local syst,proj,inc;
  inc:={seq(x[k],k=1..nops(BASE))}:
  proj:=sum(x[k]*BASE[k],k=1..nops(BASE)):
  syst:={seq(psi(proj-u,BASE[k]),k=1..nops(BASE))}:
  subs(solve(syst,inc),proj):
  end:
```

Exercice 1.

Ecrire une procédure `MatriceSchmidt(M)` renvoyant l'orthonormalisée de Gram-Schmidt de la famille des colonnes d'une matrice M inversible donnée de $\mathfrak{M}_n(\mathbb{R})$. On souhaite une matrice comme résultat.

Exercice 2.

Soit $A \in \mathscr{S}_n(\mathbb{R})$ et

$$\begin{aligned} \phi : (\mathbb{R}^n)^2 &\longrightarrow \mathbb{R} \\ (X,Y) &\longmapsto -\det \begin{pmatrix} A & X \\ {}^tY & 0 \end{pmatrix} \end{aligned}$$

1. Montrer que ϕ est une forme bilinéaire symétrique sur $\mathbb{R}^n$.

2. A quelle condition nécessaire et suffisante ϕ définit-elle un produit scalaire sur $\mathbb{R}^n$?

3. On pose

$$A = \frac{1}{12} \cdot \begin{pmatrix} 11 & 5 & -4 \\ 5 & 11 & -4 \\ -4 & -4 & 20 \end{pmatrix}$$

3.a. Montrer que ϕ est un produit scalaire.

3.b. Donner la matrice dans la base canonique de la projection orthogonale pour ϕ sur la plan d'équation $2x + y - z = 0$.

Exercice 3.

Calculer

$$\delta = \inf_{(a,b,c)\in\mathbb{R}^3} \sum_{n=0}^{+\infty} \frac{(n^3 + an^2 + bn + c)^2}{2^n}$$

3. Isométries en dimension deux et trois

On utilisera la librairie **Student[LinearAlgebra]** comportant les commandes suivantes d'algèbre linéaire et bilinéaire à destination des étudiants :

Matrices orthogonales

- **Determinant(A)** : déterminant de la matrice carrée A ;
- **Trace(A)** : trace de la matrice carrée A ;
- **IsOrthogonal(A)** : teste si la matrice A est orthogonale ;
- **RotationMatrix(θ)** : matrice de la rotation plane d'angle θ ;
- **RotationMatrix(θ,<a,b,c>)** : matrice de la rotation d'angle θ d'axe orienté vect$((a,b,c))$;
- **ReflexionMatrix(<a,b,c>)** : matrice de la réflexion d'axe vect$((a,b,c))$.

```
> with(Student[LinearAlgebra]);
```

[AddRow, AddRows, Adjoint, ApplyLinearTransformPlot, BackwardSubstitute, BandMatrix, Basis, BilinearForm, CharacteristicMatrix, CharacteristicPolynomial, ColumnDimension, ColumnSpace, CompanionMatrix, ConstantMatrix, ConstantVector, CrossProductPlot, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, EigenPlot, EigenPlotTutor, Eigenvalues, EigenvaluesTutor, Eigenvectors, EigenvectorsTutor, Equal, GaussJordanEliminationTutor, GaussianElimination, GaussianEliminationTutor, GenerateEquations, GenerateMatrix, GramSchmidt, HermitianTranspose, Id, IdentityMatrix, IntersectionBasis, InverseTutor, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, LUDecomposition, LeastSquares, LeastSquaresPlot, LinearSolve, LinearSolveTutor, LinearSystemPlot, LinearSystemPlotTutor, LinearTransformPlot, LinearTransformPlotTutor, MatrixBuilder, MinimalPolynomial, Minor, MultiplyRow, Norm, Normalize, NullSpace, Pivot, PlanePlot, ProjectionPlot, QRDecomposition, RandomMatrix, RandomVector, Rank, ReducedRowEchelonForm, ReflectionMatrix, RotationMatrix, RowDimension, RowSpace, SetDefault, SetDefaults, SumBasis, SwapRow, SwapRows, Trace, Transpose, UnitVector, VectorAngle, VectorSumPlot, ZeroMatrix, ZeroVector]

On testera ces commandes au moyen de l'exemple suivant : on vérifie que la matrice

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

est une matrice de rotation (elle est orthogonale et son déterminant vaut 1), puis on calcule son axe (en cherchant un vecteur directeur « Axe » de $\mathrm{Ker}(A - I_3)$) et son angle au signe près (ici $\pm\arccos((\mathrm{tr}(A)-1)/2) = \pm\frac{2\pi}{3}$) et on conclut que son angle vaut $-\frac{2\pi}{3}$ car son signe est celui du produit mixte $[B, AB, \text{Axe}]$ où $B \perp \text{Axe}$.

```
> A:=Matrix([[0,1,0],[0,0,1],[1,0,0]]): IsOrthogonal(A), Determinant(A);
```

true, 1

```
> arccos((Trace(A)-1)*0.5), Axe:=op(NullSpace(A-IdentityMatrix(3)));
```

$$\frac{2\pi}{3},\ \text{Axe} := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

```
> B:=Vector([-1,1,0]): Determinant(convert([B,A.B,Axe],Matrix));
```

-3

Exercice 4.

Déterminer $(a,b,c) \in \mathbb{R}^3$ pour que

$$A = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & a \\ \frac{1}{\sqrt{3}} & 0 & b \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & c \end{pmatrix}$$

soit une matrice de rotation.

4. Décomposition QR

Toute matrice M de $\mathfrak{M}_{n,p}(\mathbb{R})$ se décompose en un produit QR où Q est une matrice orthogonale et R une matrice pseudo-triangulaire supérieure [2]. Il existe plusieurs algorithmes permettant d'obtenir cette factorisation : une méthode utilisant directement le procédé d'orthonormalisation de *Gram-Schmidt* (dans le cas où $p = n = \mathrm{rg}(M)$), la méthode d'*Householder* et la méthode de *Givens*. Nous n'exposerons ici que les deux premières.

Cette décomposition est très employée en analyse numérique. Elle permet notament une résolution particulière efficace des systèmes linéaires. Soit à résoudre $MX = Y$ d'inconnue X. En décomposant M sous la forme $M = QR$, ce système équivaut à $RX = {}^t Q \cdot Y$ car Q est orthogonale. Ce nouveau système est triangulaire et très aisé à résoudre. Un des champs d'application possible de cette décomposition sera donc la méthode des moindres carrés où l'on trouve souvent des systèmes linéaires sur-déterminés, i.e. ayant plus d'équations que d'inconnues.

Au-delà de cet aspect, la décomposition QR est au fondement de la redoutablement efficace *méthode QR* de calcul approché des valeurs propres d'une matrice carrée.

4.1. Via l'algorithme de Schmidt

Considérons une matrice $M \in \mathfrak{M}_n(\mathbb{R})$ que nous supposerons de rang n. Soient $M_1, \dots, M_n$ les colonnes de M. En appliquant la méthode d'orthonormalisation de Gram-Schmidt à la famille $(M_1, \dots, M_n)$, on obtient une famile orthonormée $(Q_1, \dots, Q_n)$. De plus, pour tout entier naturel j compris entre 1 et n, on a

$$\mathrm{vect}(M_1, \dots, M_j) = \mathrm{vect}(Q_1, \dots, Q_j)$$

Ainsi, il existe des réels $r_{1,j}, r_{2,j}, \dots, r_{j,j}$ tels que

$$M_j = \sum_{i=1}^{j} r_{i,j} \cdot Q_i$$

On peut « *visualiser* » ces égalités au moyen d'un produit matriciel :

2. C'est-à-dire rectangulaire avec des coefficients $r_{i,j}$ nuls si $i > j$

$$\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ Q_1 & Q_2 & \dots & Q_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ M_1 & M_2 & \dots & M_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} = \begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,n} \\ 0 & r_{2,2} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & r_{n,n} \end{pmatrix}$$

En posant $r_{i,j} = 0$ pour $1 \leqslant j < i \leqslant n$, et $R = \left(r_{k,i}\right)_{1 \leqslant k,i \leqslant n}$, on a bien $M = Q \cdot R$ avec Q la matrice orthogonale dont les colonnes sont $Q_1, \dots, Q_n$ et R triangulaire supérieure.

On en déduit un premier algorithme de factorisation QR :

Algorithme de décomposition QR via l'orthonormalisation de Schmidt

Initialisation :
$$\begin{cases} Q_1 \leftarrow M_1 / \|M_1\| \\ R \leftarrow 0 \\ r_{1,1} \leftarrow \|M_1\| \end{cases}$$

Pour j **variant de** 2 **à** n **faire**
$$\begin{cases} Q_j \leftarrow M_j - \sum_{i=1}^{j-1} \langle M_j | Q_i \rangle \cdot Q_i \\ r_{j,j} \leftarrow \|Q_j\| \\ Q_j \leftarrow Q_j / r_{j,j} \\ \textbf{Pour } i \textbf{ variant de } 1 \textbf{ à } j-1 \textbf{ faire } r_{i,j} \leftarrow \langle M_j | Q_i \rangle \end{cases}$$

Renvoyer la matrice Q **de colonnes** $Q_1, \dots, Q_n$ **et la matrice** R

Exercice 5. *Décomposition QR par l'algorithme de Schmidt*

On reprend les notations du paragraphe précédent.

1. Ecrire une procédure `QRSchmidt(M)` renvoyant la décomposition QR d'une matrice M (donnée sous la forme d'une liste $[Q, R]$) par l'algorithme de Gram-Schmidt.

2. Que renvoie votre procédure pour les matrices suivantes :

$$M_1 = \begin{pmatrix} 1 & 1 \\ \frac{100000001}{100000000} & 1 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 1 \\ 1.00000001 & 1 \end{pmatrix}$$

Comment expliquer ce résultat ?

4.2. Méthode de Householder

Comme nous l'avons vu au travers de l'exercice précédent, la méthode de décomposition QR fondée sur l'orthonormalisation de Gram-Schmidt n'est pas numériquement stable. Voici une autre méthode, beaucoup plus stable, inventée par Householder dans les années 1950. Ce dernier est un des grands pionniers de l'analyse numérique matricielle et son algorithme figure en bonne position parmi les algorithmes qui ont révolutionné les mathématiques appliquées.

Alson Scott Householder

Pour tout vecteur colonne X non nul, on notera

$$H(X) = I_n - 2 \cdot \frac{X \cdot {}^tX}{\|X\|^2}$$

On prouve sans peine [3] que $H(X)$ est la matrice de la réflexion orthogonale d'hyperplan $(\text{vect}(X))^\perp$. Notons $C_1(M), \dots, C_n(M)$ les colonnes de la matrice M. On notant E_1 le premier vecteur de la base canonique de $\mathfrak{M}_{n,1}(\mathbb{R})$. Si $C_1(M) \neq E_1$, on a, en posant $Q_1 = H\big(C_1(M) - \|C_1(M)\| \cdot E_1\big)$,

$$Q_1 \cdot M = \begin{pmatrix} \|C_1(M)\| & \star & \dots & \star \\ 0 & & & \\ \vdots & & M_1 & \\ 0 & & & \end{pmatrix}, \quad \text{avec } M_1 \in \mathfrak{M}_{n-1,p-1}(\mathbb{R})$$

puisque Q_1 est la matrice (orthogonale) de la réflexion orthogonale d'hyperplan

$$\big(\text{vect}\,(C_1(M) - \|C_1(M)\| \cdot E_1)\big)^\perp$$

transformation géométrique qui échange les vecteurs $C_1(M)$ et $\|C_1(M)\| \cdot E_1$. Si $C_1(M) = E_1$, on obtient le même résultat avec $Q_1 = I_n$.

On procède alors de même avec la matrice M_1, de taille $n-1$, au moyen d'une matrice Q_2' orthogonale de $\mathfrak{M}_{n-1}(R)$. En posant

$$Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & Q_2' \end{pmatrix}$$

on a

$$Q_2 \cdot Q_1 \cdot M = \begin{pmatrix} \|C_1(M)\| & \star & \dots & \dots & \dots & \star \\ 0 & \|C_2(M_1)\| & \star & \dots & \dots & \star \\ \vdots & 0 & & & & \\ \vdots & \vdots & & M_2 & & \\ 0 & 0 & & & & \end{pmatrix}, \quad \text{avec } M_2 \in \mathfrak{M}_{n-2,p-2}(\mathbb{R})$$

3. en calculant l'expression de la projection orthogonale sur $\text{vect}(X)$.

En itérant ce procédé, on obtient des matrices orthogonales $Q_1, Q_2, \dots, Q_{n-1}$ telles que

$$R = Q_{n-1} \cdot Q_2 \cdots Q_1 \cdot M$$

soit pseudo-triangulaire supérieure. Et en posant $Q = Q_1 \cdot Q_2 \cdots Q_{n-1}$, on a donc $M = Q \cdot R$.

Algorithme de décomposition QR par la méthode Householder

Initialisation : $\begin{cases} R \leftarrow M \\ Q \leftarrow I_n \end{cases}$

Pour i **variant de** 1 **à** $n-1$ **faire** $\begin{cases} \text{Construire la matrice } Q_i \\ \begin{cases} R \leftarrow Q_i \cdot R \\ Q \leftarrow Q \cdot Q_i \end{cases} \end{cases}$

Renvoyer $[Q, R]$

Exercice 6. *Mise en œuvre de la méthode d'Householder de décomposition QR*

On reprend les notations du paragraphe précédent.

1. Ecrire une procédure `MatHouseholder(X)` prenant en argument $X \neq 0$ un vecteur colonne de taille $p \leqslant n$ et créant la matrice carrée de taille n suivante :

$$Y = \begin{pmatrix} I_{n-p} & 0 \\ 0 & H(X - \|X\| \cdot E) \end{pmatrix} \quad \text{où} \quad E = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathfrak{M}_{p,1}(\mathbb{R})$$

si $X \neq E$, $Y = I_n$ sinon.

2. Ecrire une procédure `QRHouseholder(M)` renvoyant la décomposition QR d'une matrice M (donnée sous la forme d'une liste $[Q, R]$) par la méthode de Householder.

3. Que renvoie votre procédure pour les matrices suivantes :

$$M_1 = \begin{pmatrix} 1 & 1 \\ \frac{100000001}{100000000} & 1 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 1 \\ 1.00000001 & 1 \end{pmatrix}$$

On fera la comparaison avec la procédure `QRSchmidt`.