Feuille TP n° 3 : Héritage

L'objectif de ce TP est d'implémenter finale de la librairie d'algèbre linéaire qui sera utilisée pour les projets qui doit permettre de réaliser les opérations standardes entres matrices et vecteurs. Une étude détaillée des différents mécanismes mis en jeu devra permettre de mettre en évidence une hiérarchie de classe : une classe mère *Tableau* de laquelle les classes matrices et vecteurs héritent naturellement.

Vous devrez apporter un soin tout particulier à la gestion de la mémoire et à la documentation de votre code.

Les données de la classe mère seront stockées de façon contiguë en mémoire.

1. En vous basant sur la classe Vecteur du TP2, implémenter une classe `Darray` contenant les attributs communs des futurs classes filles vecteurs et matrices.

   ```
   class Darray {
   private:
     int size; /* taille du tableau */
     double *data; /* données */
     bool owner; /* flag propriétaire de data */
   };
   ```

   Cette classe `Darray` utilise le `double` comme type de base et devra proposer les fonctionnalités suivantes :

   - addition, soustraction, multiplication, division par un scalaire.
   - addition, soustraction, multiplication, division termes à termes.
   - opérateur − unaire
   - `Darray view(int i1, int i2, bool copy=false);`
   - opérateur =
   - opérateur `(i)`

2. Construire une classe `Dvector` héritant publiquement de `Darray` implémentant le produit scalaire de 2 vecteurs.
3. Construire une classe `Dmatrix` de matrices (non forcément carrées) héritant publiquement de `Darray`

   ```
   class Dmatrix : public Darray {
   private:
     int m; /* nombre de lignes */
     int n; /* nombre de colonnes */
   };
   ```

   Les données de la matrice seront stockées de sorte que `M(i,j) = data[j + i*n]` (stockage ligne). Cette classe devra implémenter les fonctionnalités

   - opérateur `(i,j)`
   - opérateur =
   - produit matrice vecteur

- extraction d'une ligne d'une matrice sous la forme d'une vecteur grâce à la méthode `view`.
- extraction d'une colonne (cette fonctionnalité ne pourra pas utiliser la méthode `view` car les données d'une colonne ne sont pas stockées de manière contiguë en mémoire).

4. Implémenter la factorisation de Cholesky pour les matrices carrées symétriques et définies positives. La factorisation sera conservée dans la matrice initiale. On rappelle que la factorisation de Cholesky est donnée par

Algorithm 1 Algorithme de Cholesky

for k=1 à n **do**
  $L_{kk} = \sqrt{A_{kk} - \sum_{s=1}^{k-1} L_{ks}^2}$
  for i=k+1 à n **do**
    $l_{ik} = \left(A_{ik} - \sum_{s=1}^{k-1} L_{is} L_{ks} \right) / L_{kk}$
  end for
end for
Sortie (L_{ij})

avec A la matrice initiale (symétrique définie positive) et L une matrice triangulaire inférieure telle que $A = LL^t$.