TD n°06 : Le Pivot de Gauss

2 heures

Rédigé par Pascal Delahaye

12 décembre 2015

Récupérer le fichier `gauss.py` qui se trouve dans l'espace de partage du réseau de l'établissement.
Vous y trouverez les consignes suivantes :

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov  5 15:28:26 2014

@author: Pascal
"""

#############################################################################################

# Programmation de l'algorithme du Pivot de Gauss pour un système de Cramer

# On souhaite résoudre le système de Cramer AX=B
# La matrice A sera codée sous la forme d'une liste de listes A de taille nxn
# Le terme aij sera alors obtenu grace à la commande A[i][j] pour i et j dans [|0,n-1|]
# le vecteur B sera codé sous la forme d'une liste de valeurs : B = [b0,b1..., bn-1]
# On pourra tester les programmes suivants avec les matrices A et B suivantes :

A = [[1,0,-1],[1,2,-1],[0,2,1]]
B = [0,2,-1]

#############################################################################################

# I] Programmation des opérations élémentaires

# 1) Recherche d'un pivot

# Compléter la fonction suivante qui dans une colonne j d'une matrice A, renvoie
# l'indice du premier coefficient aij non nul pour i>=j. Ce coefficient sera le pivot.
# Le système étant de Cramer, on est assuré qu'un tel coeffient existe bien

def pivot(A,j):
    n = len(A)                # Ne sert à rien !!
```

```
    i=j
    while A[i][j] == 0:
        ....

pivot(A,0)

# 2) Echange de lignes : Li <-> Lj

# Compléter la fonction suivante qui échange les lignes Li et Lj d'une matrice A
# Attention, l'indexation des lignes et colonnes commence à 0

def echange_lignes (A,i,j):
    n = len(A)
    for k in range(...) :
        ...

echange_lignes(A,0,2)

# 3) Transvection : Li <- Li + mu.Lj

# Compléter la fonction suivante qui dans une matrice A, remplace la ligne Li par la ligne Li + mu.Lj

def transvection(A,i,j,mu):
    n = len(A)
    for k in range(...) :
...

transvection(A,0,1,2)

#########################################################################################################

# II] Programmation de la résolution du système


# 1) ETAPE 1 : Triangularisation du système

# Complèter la procédure suivante qui transforme le système AX = B en un système triangulaire
# On pensera à bien effectuer sur le vecteur B les mêmes opérations d'échange et de transvection
# que sur la matrice A

def triangularisation(A,B):
    n = len(A)
    for i in range(0,n-1):          # j parcourt ici toutes les lignes de 0 à n-2
        k = pivot(...)
        ...                         # on échange les lignes Li et Lk puis
                                    # on annule tous les coefficients sous a[i][i]

triangularisation(A,B)


# 2) ETAPE 2 : Résolution du système triangulaire

# Construire la fonction qui, à partir d'un système triangulaire, renvoie le vecteur solution.

def soltrig(A,B):
    n = ln(A)
    l =[] # l est la liste dans lesquelles nous allons stocker les solutions
    for j in range(0,n-1):      # j représente l'indice de la colonne traitée
...

soltrig(A,B)
```

```
# 3) ETAPE 3 : Programme de résolution globale

# A l'aide des procédures précédentes, construire une fonction de résolution du système AX = B
# par la méthode de Gauss.

#############################################################################################################
```