esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

Examen

Semestre : 1 ☐ 2 ☒
Session: Principale ☒ Rattrapage ☐

Module: Analyse Numérique
Enseignant(s): Équipe AN
Classe(s): 3A & 3B
Documents autorisés: OUI ☐ NON ☒ Nombre de pages: 4
Calculatrice autorisée: OUI ☒ NON ☐ Internet autorisée: OUI ☐ NON ☒
Date: 09 Janvier 2025 Heure: 11h00 Durée : 1h30 min

NB : Vous êtes appelés **à écrire que quatre chiffres après la virgule** pour tous résultats trouvés. Pour les questions pratiques, notez bien que **cette feuille d'examen n'est pas à rendre**. Veuillez donc **compléter les codes concernés** dans la **feuille de réponse**.

Exercice 1 : (6.5 points)

Le **nombre d'or**, souvent noté φ, est une constante mathématique remarquable qui intervient dans divers domaines tels que les arts, la nature, l'architecture, et la géométrie. Il est défini comme étant l'unique solution positive de l'équation quadratique (E) donnée par :

$$(E) \quad x^2 - x - 1 = 0 \qquad \text{sur } [\frac{3}{2}, 2].$$

1) On s'intéresse à approcher numériquement la valeur de φ par la méthode de Newton.
   a) **(1 point)** Donner le schéma itératif de la méthode de Newton associé à l'équation (E).
   b) **(1.5 points)** Étudier la convergence de la méthode de Newton pour la résolution de l'equation (E).
   c) **(1.5 points)** Calculer les trois premiers itérés par la méthode de Newton.
2) On souhaite maintenant approcher la racine φ de l'équation (E) en utilisant le schéma itératif de point fixe (Sx) suivant :

$$(Sx) \ : \begin{cases} x_0 \in [\frac{3}{2}, 2], \\ x_{n+1} = 1 + \dfrac{1}{x_n}, \quad \forall n \geq 0. \end{cases}$$

   a) **(1.5 points)** Justifier la convergence du schéma (Sx) pour résoudre l'équation (E).
   b) **(1 point)** Déterminer le nombre des itérations suffisant pour approcher φ à 10^{-5} près, par le schéma itératif (Sx).

Exercice 2 (4.5 points)

On réalise une expérience chimique pour mesurer la concentration $C = (c_i)_{1 \leq i \leq 3}$ d'un produit chimique à différentes températures $T = (t_i)_{1 \leq i \leq 3}$. Les valeurs obtenues sont récapitulées dans le tableau suivant :

Température T_i (°C)	Concentration C_i (mol/L)
30	0.02
40	0.05
50	0.10

Tableau 1 – Tableau des concentrations mesurées à différentes températures

Partie 1 : Étude du polynôme d'interpolation de Newton.

1) **(0.5 point)** Donner la condition nécessaire pour assurer l'existence et l'unicité du polynôme d'interpolation de Newton P_2 pour les ensembles de données $C = (c_i)_{1\leq i\leq 3}$ et $T = (t_i)_{1\leq i\leq 3}$.
2) **(1 point)** En utilisant la méthode d'interpolation de Newton, montrer que :

$$P_2(X) = 0.05 - 0.004X + 0.0001X^2.$$

3) **(1 point)** Déduire la valeur approchée de la température correspondant à une concentration de $0,09\, mol/L$.

Partie 2 : Approximation polynomiale des données expérimentales.

1) **(1.5 points)** Trouver les coefficients λ_0 et λ_1 de la droite de régression d'équation $y = \lambda_0 + \lambda_1 x$, au sens des moindres carrées, qui ajuste au mieux les points donnés.
2) **(0.5 point)** Déduire la concentration du produit chimique à la température 45°C.

Exercice 3 : (9 points)

Pour étudier la distribution de la chaleur dans une chambre, on est amené à résoudre un système d'équations linéaires (S_n) de la forme $AX = b$, où n représente le nombre de points de discrétisation.
Ce système modélise la propagation de la chaleur dans un réseau discret de points, où chaque équation représente l'équilibre thermique à un point donné en fonction de ses voisins.
La matrice A d'ordre n ainsi que les vecteurs colonnes X et b sont définis comme suit :

$$A = \begin{pmatrix} 3 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 3 & -1 & \ddots & \ddots & 0 \\ 0 & -1 & 3 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 3 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 2 \\ 2 \\ \vdots \\ \vdots \\ 2 \\ 2 \end{pmatrix}$$

Où A est une matrice tridiagonale et symétrique.

1) On suppose que $n = 3$
   a) **(0.5 point)** Montrer que (S_3) admet une unique solution.
   b) **(1.5 points)** En utilisant la méthode de pivot de Gauss, montrer que $X = \begin{pmatrix} \frac{8}{7} \\ \frac{10}{7} \\ \frac{8}{7} \end{pmatrix}$ est la solution de (S_3).
2) On suppose que n est assez grand et on se propose de résoudre numériquement le système (S_n) .
   Pour obtenir une solution approchée du système linéaire (S_n), on considère le schéma itératif (SI) suivant :

$$(SI) \quad \begin{cases} \text{Pour } X^{(0)} \text{ donnée,} \\ X^{(k+1)} = (D - \omega E)^{-1} \Big[\left((1-\omega)D + \omega F\right) X^{(k)} + \omega b \Big], \quad \forall k \geq 0, \end{cases}$$

   où $0 < \omega < 2$ et $A = D - E - F$ avec :

$$A = \underbrace{\begin{pmatrix} 3 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 3 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 3 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 0 & 3 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 1 & 0 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{pmatrix}}_{E} - \underbrace{\begin{pmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 0 & 0 \end{pmatrix}}_{F}$$

a) **(1.5 points)** Écrire une fonction nommée `système(n)` prenant en entrée n la dimension du système (S_n), et qui renvoie A la matrice tridiagonale et b le second membre, associés au système (S_n).

[ ]:
```
import .....................
def .....................









    return ......................
```

b) **(1.5 points)** Pour pouvoir appliquer le schéma (SI), il suffit de s'assurer que la matrice A soit à diagonale strictement dominante. Compléter la fonction nommée `diag_dominante(A)` prenant en entrée la matrice A, et qui retourne la booléenne `True` si la matrice A est à diagonale strictement dominante et `False` sinon.

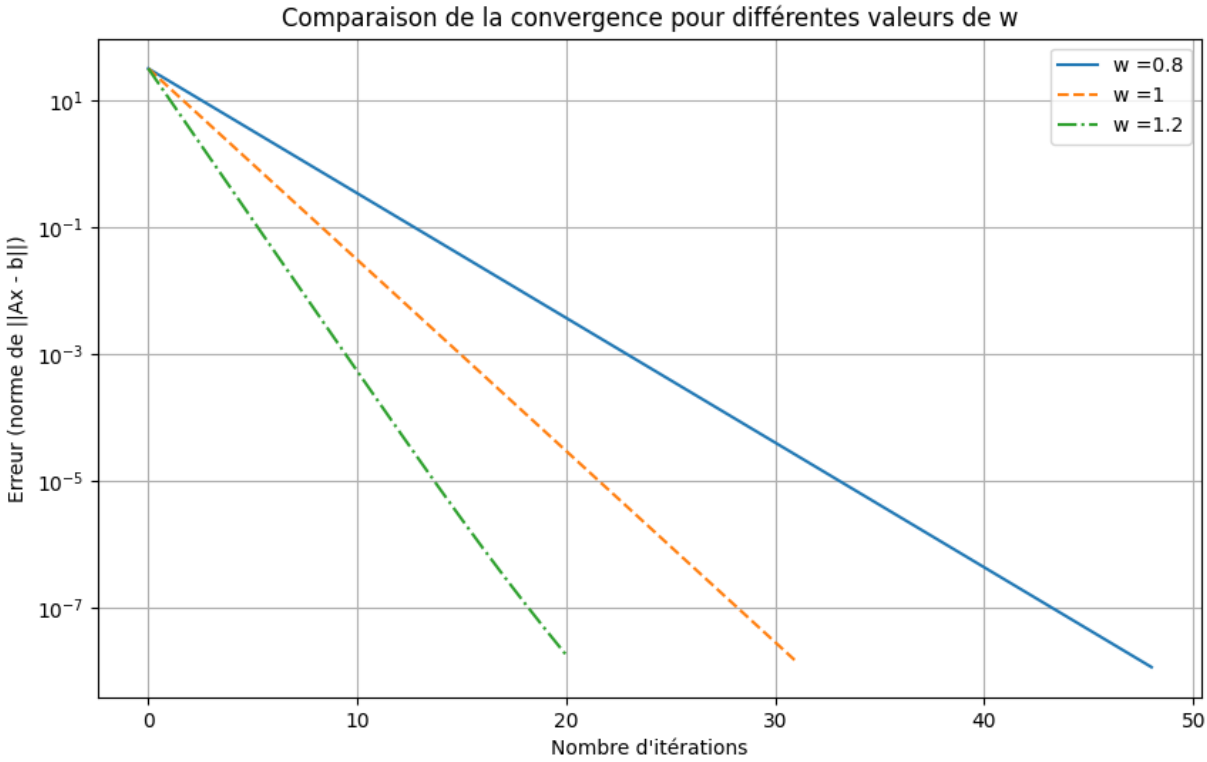
[ ]:
```
def diag_dominante(A):
    n=......................
    etat=True
    i=0
    while ....................................
        etat = abs(A[i,i]) > ..............................
        ......................
    return etat
```

c) **(2 points)** Compléter la fonction `schema(A,b,X0, w, epsilon)` prenant en entrée la matrice `A`, le second membre `b`, la valeur initiale `X0`, la constante `w` et la tolérance `epsilon`, et qui retourne une liste des erreurs $\|A \cdot X_k - b\|$ pour chaque itération (où $\|.\|$ désigne la norme euclidienne), ainsi que le nombre d'itérations effectuées pour atteindre la convergence par le schéma (SI) sachant que $0 < w < 2$ et que A soit à diagonale strictement dominante. Sinon, un message d'erreur doit être renvoyé.

[ ]:
```
def schema(A,b, X0, w, epsilon):
  Etat = ......................
  if Etat==True and ......................... :
    ERROR = [] # initilisation de la liste des erreurs
    k = 0
    D = ......................
    E = ......................
    F = ......................
    while np.linalg.norm(A.dot(X0) - b) > epsilon:
      erreur = np.linalg.norm(A.dot(X0) - b)
      ERROR.append(erreur)
      X0 = ................................................
      k += 1
    return ERROR, k
  else:
    return "choix incorrecte de w ou A n'est pas à diagonale strict dominante"
```

d) **(1.5 points)** Compléter le code suivant par les instructions nécessaires pour tracer la figure (ci-dessous) des erreurs commises en norme euclidienne en fonction du nombre d'itérations, pour résoudre le système (S_{1000}) pour différentes valeurs de w.

[ ]:
```
n=1000
X0 = np.ones((n, 1))
epsilon = 10**-8
W = [ 0.8,1, 1.2]
Line = ['-', '--', '-.']
k=0
import ......................
plt.figure(figsize=(10, 6))
for w in W:
  [erreurs, iterations] = ......................
  plt.plot(........................, erreurs,label='w =' + str(w),linestyle=Line[k])
  k+=1
  plt.yscale('log')
  plt.xlabel("Nombre d'itérations")
  plt.ylabel('Erreur (norme de ||Ax - b||)')
  ..............('Comparaison de la convergence pour différentes valeurs de w')
  plt.legend()
  ......................     # ajouter la grille
plt.show()
```



e) **(0.5 point)** Interpréter graphiquement les résultats obtenus.

.....

.....