Problème 1 — Equations Différentielles Ordinaires (11.4 points)

On considère le problème de Cauchy suivant :

$$\begin{cases} y_1'(t) = -\frac{4}{3}y_1(t) + \frac{1}{3}y_2(t) + \frac{1}{3}e^{4-t}, & t \in [0,5], \\ y_2'(t) = 3y_1(t) - 4y_2(t), \\ y_1(0) = 8, \\ y_2(0) = 1, \end{cases} \tag{1}$$

(a) Cocher les bonnes réponses parmi les choix proposés. (Bonne réponse 0.2 points, mauvaise -0.2, pas de réponse 0.)

Méthode	explicite	implicite	ordre 1	ordre 2	ordre 3	cond. stable	incond. stable
Heun	☐	☐	☐	☐	☐	☐	☐
Euler Retrograde	☐	☐	☐	☐	☐	☐	☐
Euler Progressif	☐	☐	☐	☐	☐	☐	☐
Crank–Nicholson	☐	☐	☐	☐	☐	☐	☐

(b) Ecrire le problème de Cauchy (1) sous forme vectorielle en introduisant la variable $\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}$.

☐ ☐ ☐ ☐ ☐ *Réservé au correcteur*

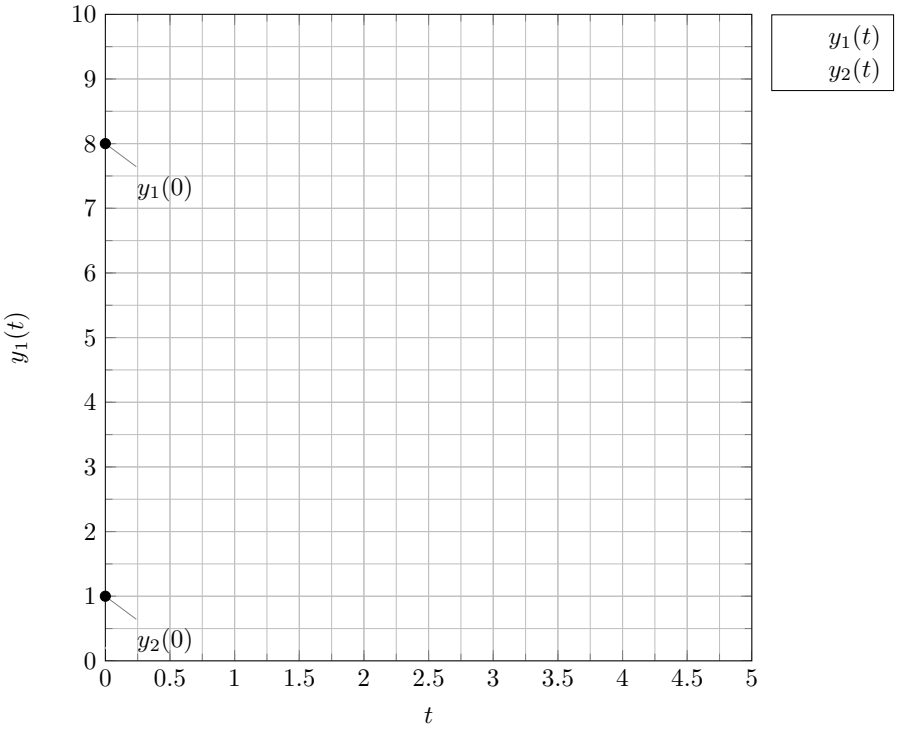
(c) On veut approcher la solution $\mathbf{y}(t)$ de (1) par la méthode d'Euler Rétrograde. Ecrire la méthode pour un problème de Cauchy général ainsi que pour approcher la solution de (1). Indiquer quelles sont les propriétés de cette méthode.

Réservé au correcteur

(d) Utiliser la fonction Matlab `backwardEuler.m` (sur votre ordinateur) pour dessiner une esquisse de l'approximation de $y_1(t)$ et $y_2(t)$ pour $h = 0.25$. *Reporter les commandes Matlab utilisées.*

Réservé au correcteur

Dessiner ici les graphes des approximations de $y_1(t)$ et $y_2(t)$ obtenues, inclure la légende :



(e) Déterminer, si nécessaire, pour quelles valeurs de h la condition de stabilité (contrôle des perturbations) pour la méthode d'Euler Progressive est satisfaite. Vous pouvez utiliser Matlab. Vérifier si la méthode est stable pour $h = 0.25$. *Reporter aussi les éventuelles commandes Matlab utilisées.*

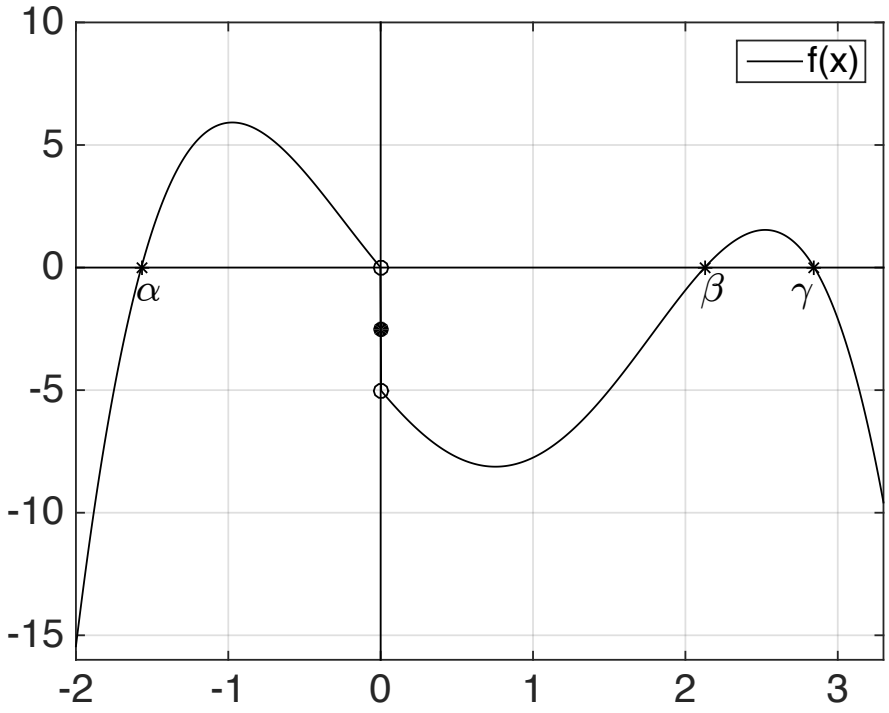
Réservé au correcteur

(f) En sachant que $y_1(0.5) = 9.743790310568556$ (voir aussi la fonction `p1sol.m`), calculer de façon approximative l'ordre de convergence, par rapport à la première composante $y_1(0.5)$ en $t = 0.5$ seulement, de la méthode implémentée dans `Hom.m`. *Reporter aussi les commandes Matlab utilisées.*

Réservé au correcteur

Problème 2 — Equations Non-Linéaires (9 points)

Soit $f :\to \mathbb{R}$ la fonction représentée dans le graphique suivant. On observe qu'elle a trois zéros α, β et γ. La fonction, sa première et sa deuxième dérivée peuvent être évaluées en Matlab à l'aide des fonctions `p2f.m`, respectivement `p2df.m` et `p2d2f.m` (utiliser aussi la commande `help`).



(a) On veut utiliser la méthode de bissection pour calculer une approximation des zéros de f. Indiquer pour quels sous-intervalles la méthode de bissection peut effectivement être utilisée. En chaque intervalle où il est possible d'utiliser la méthode de bissection, estimer le nombre minimal d'itérations nécessaires pour trouver un zéro avec une erreur inférieure à 10^{-6}. (Bonne réponse 0.4 points, mauvaise 0.)

	0-1	4-5	16-17	18-19	20-21	22-23	bisection pas utilisable
$[-2, 3]$	☐	☐	☐	☐	☐	☐	☐
$[-2, -0.5]$	☐	☐	☐	☐	☐	☐	☐
$[1, 3]$	☐	☐	☐	☐	☐	☐	☐
$[2.1, 2.2]$	☐	☐	☐	☐	☐	☐	☐
$[-1, 1.5]$	☐	☐	☐	☐	☐	☐	☐

(b) Soit $q \neq 0$ un nombre réel. Pour calculer une approximation du zéro $\alpha < 0$ de f, on veut utiliser la méthode de point fixe associée à la fonction $\phi(x) = x - f(x)/q$. Quelle est la condition sur q pour que la méthode converge localement vers α ? Y a-t-il un choix optimal ? Dans ce cas, quel est l'ordre de convergence de la méthode ?

Justifiez vos réponses.

Réservé au correcteur

Réservé au correcteur

(c) Calculer l'approximation de α en utilisant la méthode de point fixe décrite précédemment avec une valeur q de votre choix :

- Implémenter dans `rootFinder.m` la méthode de point fixe décrite pour trouver un zéro d'une fonction. Ne pas oublier de mettre des commentaires dans le code.
- Choisir $x_\alpha^{(0)} = -2$ comme valeur initiale pour `rootFinder` avec une valeur q de votre choix et approximer α en utilisant respectivement 3, 4 et 5 itérations.

Réporter les commandes Matlab utilisées, les résultats obtenus, puis les commenter.

Réporter ici la fonction ***`rootFinder`****, avec les commentaires opportuns.*

Réservé au correcteur

```
function [x, r] = rootFinder( F, q, x0, niter )
% [x,r] = rootFinder(F, q, x0, niter)
% rootFinder cherche le zero d une equation non-lineaire.
%
% Fonction Matlab qui cherche le zero X de la
% fonction continue F le plus proche de x0 en
% utilisant un nombre d iterations egal a niter.
%
% INPUT:
%   - F:     fonction du probleme
%   - q:     nombre reel different de zero
%   - x0:    approximation initiale du zero
%   - niter: nombre d iterations
%
% OUTPUT:
%   - x: zero calule de la methode
%   - r: residue, evalue comme abs(F(x));
%
% EXEMPLE:
%  [x, r] = rootFinder( @p2f , 1, 2, 3);
```

Réporter les commandes Matlab utilisées et commenter les résultats obtenus.

Réservé au correcteur

Problème 3 — Systèmes Linéaires (11 points)

On considère les matrices suivantes :

$$A = \begin{pmatrix} 3 & 1 & -2 & 0.5 \\ 1 & 4 & 0 & 0.5 \\ -2 & 0 & 7 & 1 \\ 0.5 & 0.5 & 1 & 2.5 \end{pmatrix} \quad B = \begin{pmatrix} 3 & 1 & -2 & 0.5 \\ 1 & 0 & 0 & 0.5 \\ -2 & 0 & 7 & 1 \\ 0.5 & 0.5 & 1 & 2.5 \end{pmatrix} \quad C = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ -2 & 0 & 7 & 0 \\ 0.5 & 0.5 & 1 & 2.5 \end{pmatrix} \quad D = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 7 & 1 \\ 0 & 0 & 1 & 2.5 \end{pmatrix}$$

Pour un vecteur quelconque $\mathbf{b} \in \mathbb{R}^4$, on aimerait résoudre $A\mathbf{x} = \mathbf{b}$, $B\mathbf{x} = \mathbf{b}$, etc. Pour ce faire, on se pose la question d'inversibilité des matrices, si elles sont symétriques positives définies, ou si différentes méthodes itératives vont converger vers la solution pour tout choix initial $\mathbf{x}^{(0)}$ (ou pas).

(a) A cette fin, remplissez le tableau suivant. Vous pouvez utiliser Matlab, il ne faut pas reporter les commandes. (Bonne réponse 0.2 points, mauvaise 0.)

	la matrice est inversible		symétrique définie positive		Jacobi converge[1]		Gauss-Seidel converge[1]		la méthode du Gradient converge[1]	
	oui	non	oui	non	oui	non	oui	non	oui	non
A	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
B	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
C	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
D	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

[1] Selon un critère connu.

(b) Ecrire la matrice d'itération de la méthode de Richardson préconditionnée stationnaire pour résoudre le système linéaire $A\mathbf{x} = \mathbf{b}$ pour $\mathbf{b} \in \mathbb{R}^4$ en utilisant le préconditionneur $P = D$ et $\alpha_k = \alpha_{opt}$. Dire si la méthode converge et justifier la réponse en faisant un lien avec la théorie.
Si vous utilisez Matlab, reportez aussi les commandes.

☐☐☐☐☐ ☐☐☐☐☐ *Réservé au correcteur*

☐☐☐☐☐ ☐☐☐☐☐ *Réservé au correcteur*

(c) Suite aux résultats obtenus au point précédent et en supposant que $\|\mathbf{x}^{(0)} - \mathbf{x}\|_A \approx 1.4$, quel est le nombre d'itérations théoriques attendues afin d'obtenir une erreur dans la même norme inférieure à 10^{-6} ? Que peut-on dire de la méthode de Jacobi appliquée au même problème (si $\|\mathbf{x}^{(0)} - \mathbf{x}\| \approx 1.7$) ?

Réservé au correcteur

Problème 4 — Interpolation (10 points)

(a) Expliquer avec vos propres mots l'interpolation quadratique par morceaux d'une fonction $f : [a, b] \to \mathbb{R}$.

Réservé au correcteur

(b) On veut implémenter l'interpolation quadratique par morceaux pour une fonction $f : [a, b] \to \mathbb{R}$. Une partie de l'implémentation est faite dans `interp_morceaux_2.m`, cf. `help`. Elle fait appel à une fonction supplémentaire `lagrange_interp_2.m` dont la structure est donnée ci-dessous. Compléter le code de cette fonction sans oublier de commenter le code.

(voir page suivante)

Reporter la fonction ***lagrange_interp_2.m*** *ici.*

Réservé au correcteur

```
function [y] = lagrange_interp_2(nodes, f, x)
% y = lagrange_interp_2(nodes, f, x)
% Fonction Matlab qui implemente l interpolation
% de Lagrange de degre 2 aux noeuds nodes d une fonction f
% au(x) point(s) x.
%
% Input:
%     — nodes: vecteur ligne avec les valeurs des
%              3 noeuds d interpolation dans l intervalle
%     — f:     fonction que l on veut interpoler
%     — x:     point ou vecteur de points ou l on veut evaluer
%              le polynome d interpolation
% Output:
%     — y: evaluation du polynome d interpolation de
%          Lagrange de degre 2 de f au(x) point(s) x
%
% EXEMPLE
%   nodes = [0.1 0.2 0.3];
%   f     = @(x) cos(pi*x);
%   x     = 0.25;
%   y     = lagrange_interp_2(nodes, f, x)
% Dans cet exemple, le resultat devrait etre y=0.7083
```

(c) On considère l'interpolation quadratique par morceaux de deux fonctions f et g sur l'intervalle $[0, 2\pi]$ avec 10 sous-intervalles de même longueur. Les erreurs d'interpolation sont alors bornées par : (cocher la valeur la plus restrictive, bonne réponse 1 point, mauvaise 0.)

	0	1	π	3	0.018	0.707	0.875	2	$\frac{\pi^2}{4}$	$\frac{\pi^3}{96}$	$\frac{\pi^4}{24}$
Si $f(x) = sin\left(\frac{3x}{\pi}\right)$,	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Si $g(x) = 3x^2 - 2x + 1$,	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

(d) On considère les mesures suivantes :

x	2.01	2.16	2.34	2.45	2.56	2.67	2.81	2.94
$f(x)$	1.48	1.1	0.81	0.61	0.53	0.43	0.28	0.26

Des biologistes présument que le procédé est réglé par une loi de type $f(x) = C\, e^{-ax}$.

Déterminer les valeurs des constantes C et a à l'aide de la fonction Matlab `leastSquares.m` donnée qui implémente la méthode des moindres carrés (voir aussi le help de la fonction).

Un autre biologiste propose un modèle plus complet de type $g(x) = C\, e^{-ax}\, e^{bx^2}$. Vérifier aussi cette hypothèse et commenter le résultat.

Reporter aussi vos commandes Matlab.

☐☐☐☐☐ ☐☐☐☐☐ ☐☐☐☐☐ *Réservé au correcteur*

(Espace supplémentaire. Indiquer clairement quand un exercice continue ici.)

Nom: **Abbet Marie** SCIPER: **227392**

ODE

$$-\lambda_{max} \leq \frac{\partial f}{\partial x}(t,x) \leq -\lambda_{min}$$

$$h < \frac{2}{\max_{j=1,\dots,p} |\lambda_j|} = \frac{2}{\rho(A)} \ ,$$

$$\forall n = 0, \dots, N_h \quad |u_n - y(t_n)| \leq C(h)$$

$$|y(t_n) - u_n| \leq \frac{e^{Lt_n} - 1}{2L} \max_{t\in[0,T]} |y''(t)|\, h,$$

Lemma Soit E_n une suite de nombres positifs telle que $E_0 = 0$ et $E_{n+1} \leq (1+\delta)E_n + K$ avec $\delta > 0$ et $K > 0$. Alors

$$E_n \leq \frac{e^{\delta n} - 1}{\delta} K$$

Lemma Soit E_n une suite de nombres positifs telle que $E_0 = 0$ et $(1-\delta)E_{n+1} \leq (1+\delta)E_n + K$ avec $0 < \delta < \frac{1}{2}$, $K > 0$. Alors

$$E_n \leq \frac{e^{4\delta n} - 1}{2\delta} K.$$

$$u_{n+1} - u_n = \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right]$$

$$... = \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_n + hf(t_n, u_n))\right]$$

Linear Systems

$$B = P^{-1}(P - A) = I - P^{-1}A \quad , \quad \mathbf{g} = P^{-1}\mathbf{b}.$$

$$P(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \alpha_k \mathbf{r}^{(k)}$$

$$\alpha_k = \alpha_{opt} = \frac{2}{\lambda_{min}(P^{-1}A) + \lambda_{max}(P^{-1}A)}$$

$$\begin{aligned} P\mathbf{z}^{(k)} &= \mathbf{r}^{(k)} \\ \alpha_k &= \frac{(\mathbf{z}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{z}^{(k)})^T A \mathbf{z}^{(k)}} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)} \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k A \mathbf{z}^{(k)} \end{aligned}$$

$$\|\mathbf{x}^{(k)} - \mathbf{x}\|_A \leq \left(\frac{K(P^{-1}A) - 1}{K(P^{-1}A) + 1}\right)^k \|\mathbf{x}^{(0)} - \mathbf{x}\|_A$$

$$\frac{\|\mathbf{x}^{(k)} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq K(P^{-1}A) \frac{\|P^{-1}\mathbf{r}^{(k)}\|}{\|P^{-1}\mathbf{b}\|}$$

Interpolation

$$\varphi_k(x) = \prod_{j=0, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)}.$$

$$\max_{x\in I} |E_n f(x)| \leq \frac{1}{4(n+1)} (h)^{n+1} \max_{x\in I} |f^{(n+1)}(x)|,$$

$$\max_{x\in I} |\ E_1^H f(x)\ | \leq \frac{H^2}{8} \max_{x\in I} |f''(x)|.$$

$$\max_{x\in I} |\ E_n^H f(x)\ | \leq \frac{H^{n+1}}{4(n+1)} \max_{x\in I} |f^{(n+1)}(x)|\,.$$