

Rappels sur quelques méthodes numériques

L'analyse numérique a commencé bien avant la conception des ordinateurs et leur utilisation quotidienne que nous connaissons aujourd'hui. Les premières méthodes ont été développées pour essayer de trouver des moyens rapides et efficaces de s'attaquer à des problèmes soit fastidieux (ممل) à résoudre à cause de leur grande dimension (systèmes à plusieurs dizaines d'équations par exemple), soit parce qu'il n'existe pas solutions explicites connues même pour certaines équations assez simples en apparence.

Dès que les premiers ordinateurs sont apparus, ce domaine des mathématiques a pris son envol et continue encore à se développer de façon très soutenue. Les applications extraordinairement nombreuses sont entrées dans notre vie quotidienne directement ou indirectement. Nous les utilisons désormais sans nous en rendre compte mais surtout en ignorant la plupart du temps toute la théorie, l'expertise, le développement des compétences et l'ingéniosité des chercheurs pour en arriver là. Nous pouvons téléphoner, communiquer par satellite, faire des recherches sur internet, regarder des films où plus rien n'est réel sur l'écran, améliorer la sécurité des voitures, des trains, des avions, connaître le temps qu'il fera une semaine à l'avance,...et ce n'est qu'une infime partie de ce que l'on peut faire. Le but de ce cours est d'initier aux bases de l'analyse numérique en espérant qu'elles éveilleront de l'intérêt, de la curiosité et pourquoi pas une vocation.

1.1 Résolution des systèmes d'équations linéaires

La méthode itérative contrairement à d'autres à l'avantage de ne pas avoir besoin de garder en mémoire la totalité d'une matrice de très grande taille gourmande en capacités mémoire. Cette méthode permet de garder en mémoire que les coefficients non nuls d'une matrice de grande taille. Cependant, le succès de calcul n'est pas assuré quelque soit la matrice, certaines conditions sont nécessaire afin d'obtenir un résultat convergent, ce que nous allons voir dans ce document au travers des méthodes de Jacobi et de Gauss-Seidel.

$$Ax = b \quad (1.1)$$

L'idée est de déduire un schéma itératif de la décomposition de A sous la forme $A = M - N$ où M est une matrice inversible.

$$\begin{aligned}(M - N)x &= b \Leftrightarrow Mx = Nx + b \\ \Leftrightarrow M^{-1}Mx &= M^{-1}\underbrace{Nx}_I + M^{-1}b \\ \Rightarrow x &= M^{-1}Nx + M^{-1}b\end{aligned}$$

qui définit une équation de point fixe.

On considère alors le schéma itératif associé :

$$x^{(k+1)} = M^{-1}Nx^k + M^{-1}b \quad (1.2)$$

avec $x^{(0)} \in \mathbb{K}^n$ (vecteur de départ). L'algorithme est initialisé par un vecteur arbitraire

$x^{(0)} = (x_1^0, x_2^0, x_3^0, \dots, x_n^0)$ et s'arrête quand $\forall i \in \mathbb{N}, |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$ pour un ε donné.

Selon les choix des matrices M et N on a différentes méthodes itératives. Le point de départ de chacune de ces méthodes est l'unique décomposition de la matrice $A = (a_{i,j})_{1 \leq i,j \leq n}$ sous la forme $A = D - (E + F)$ avec :

- $D = (d_{i,j})_{1 \leq i,j \leq n}$ diagonale, telle que $d_{i,j} = a_{i,j}$ et $d_{i,j} = 0$ pour $i \neq j$;
- $-E = (e_{i,j})_{1 \leq i,j \leq n}$ triangulaire inférieure **stricte** telle que $e_{i,j} = a_{i,j}$ si $i > j$ et $e_{i,j} = 0$ si $i \leq j$;
- $-F = (f_{i,j})_{1 \leq i,j \leq n}$ triangulaire supérieure **stricte**, telle que $f_{i,j} = a_{i,j}$ si $i < j$ et $f_{i,j} = 0$ si $i \geq j$;

Exemple: Considérons la matrice

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}$$

La décomposition de A sous la forme $A = D - (E + F)$ décrite ci-dessus s'écrit alors :

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix} = \underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}}_D - \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ -1 & -1 & 0 \end{pmatrix}}_{-E} - \underbrace{\begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}}_{-F}$$

a) Méthode de Jacobi

On suppose que A est une matrice inversible dont aucun élément de la diagonale est nul ($a_{ii} \neq 0, \forall i$). Cette méthode consiste à isoler le coefficient de la diagonale de chaque ligne du système, si l'un des coefficients diagonaux est nul, il est parfois possible de permuter certaines lignes pour éviter cette situation.

On considère un système linéaire suivant :

$$Ax = b, \text{ avec } A \text{ inversible.}$$

On pose :

$$A = M - N, \text{ avec } M = D \text{ et } N = (E + F); \text{ sachant que } D \text{ est inversible.}$$

Le schéma itératif s'écrit alors :

$$x^{(k+1)} = D^{-1}(E + F)x^k + D^{-1}b \quad (1.3)$$

avec $x^{(0)} \in \mathbb{K}^n$

La matrice $B_J = D^{-1}(E + F)$ est appelée matrice de Jacobi.

Algorithme de Jacobi

$$\left\{ \begin{array}{l} x^{(0)} \text{ donné} \\ x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i, j=1}^n a_{ij} x_j^{(k)} \right) \end{array} \right. \quad \forall i = 1, \dots, n.$$

Cet algorithme nécessite $a_{ii} \neq 0$ pour $i = 1, \dots, n$

Explicitement, on obtient :

$$\begin{aligned} a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \dots - a_{1n}x_n^{(k)} + b_1 \\ a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k)} - a_{23}x_3^{(k)} \dots - a_{2n}x_n^{(k)} + b_2 \\ &\vdots \\ a_{nn}x_n^{(k+1)} &= -a_{n1}x_1^{(k)} - \dots - a_{nn-1}x_{n-1}^{(k)} + b_n \end{aligned}$$

Test de convergence

La méthode ne converge pas toujours. On démontre que si A est une matrice définie positive, la méthode itérative converge. De même, si A est une matrice diagonalement dominante, c'est-à-dire si

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad (1.4)$$

alors la méthode de Jacobi converge. Par conséquent, on peut avoir intérêt à réarranger les termes de A de façon à mettre A sous la forme d'une matrice dont les éléments diagonaux sont les plus grands possibles.

b) Méthode de Gauss-Seidel

Comme pour la méthode de Jacobi, on suppose cette fois que la matrice D est une matrice inversible dont aucun élément de la diagonale est nul ($a_{ii} \neq 0, \forall i$). Cette méthode consiste à isoler le coefficient de la diagonale de chaque ligne du système, si l'un des coefficients diagonaux est nul, comme pour la méthode de Jacobi, il est parfois possible de permuter certaines lignes pour éviter cette situation.

On considère un système linéaire suivant :

$$Ax = b, \text{ avec } A \text{ inversible.}$$

On pose :

$$A = M - N, \text{ avec } M = D - E \text{ et } N = F; \text{ avec } (D - E) \text{ inversible}$$

Le schéma itératif s'écrit alors :

$$x^{(k+1)} = (D - E)^{-1} F x^{(k)} + (D - E)^{-1} b$$

avec $x^{(0)} \in \mathbb{K}^n$ (vecteur de départ).

La matrice $B_{GS} = (D - E)^{-1} F$ est appelée matrice de Gauss-Seidel. A chaque pas, on calcule

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^i a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right) \quad (1.5)$$

Algorithme de Gauss-Seidel

$$\begin{cases} x^{(0)} \text{ donné} \\ x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right) \quad \forall i = 1, \dots, n. \end{cases}$$

Cet algorithme nécessite $a_{ii} \neq 0$ pour $i = 1, \dots, n$

Explicitement, on obtient :

$$\begin{aligned} a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \dots - a_{1n}x_n^{(k)} + b_1 \\ a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} \dots - a_{2n}x_n^{(k)} + b_2 \\ &\vdots \\ a_{ii}x_i^{(k+1)} &= -a_{i1}x_1^{(k+1)} - a_{ii-1}x_{i-1}^{(k+1)} - a_{ii+1}x_{i+1}^{(k)} \dots - a_{in}x_n^{(k)} + b_i \\ &\vdots \\ a_{nn}x_n^{(k+1)} &= -a_{n1}x_1^{(k+1)} - \dots - a_{nn-1}x_{n-1}^{(k+1)} + b_n \end{aligned}$$

1.2 Résolution des systèmes d'équations non linéaires

a) Méthode de Newton-Raphson

La méthode de Newton (ou Newton-Raphson) consiste à prendre pour x_{k+1} la racine du développement de Taylor de premier ordre autour de $x^{(k)}$; cela revient, pour autant que $f'(x^{(k)}) \neq 0$, à déterminer $x^{(k+1)}$ qui satisfait :

$$f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = 0$$

Et donc

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \quad (1.6)$$

Critère de convergence

Soit une fonction f définie sur $[a, b]$ telle que :

- i. $f(a)f(b) < 0$
- ii. $f'(x)$ et $f''(x)$ sont non nulles et gardent un signe constant sur l'intervalle donné.

Algorithme de Newton

(entrée : $f, f', x^{(0)}$, Sortie : $x^{(k+1)}$)
 Répéter jusqu'à l'arrêt
 Pour autant que $f'(x^{(k)}) \neq 0$

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

Critère d'arrêt

En général, pour toutes les méthodes étudiées, on peut utiliser deux critères d'arrêt différents : les itérations s'achèvent dès que :

$$|x_i^{(k+1)} - x_i^{(k)}| < \varepsilon \quad (\text{Contrôle de l'incrément})$$

Si non

$$|f(x^{(k)})| < \varepsilon \quad (\text{Contrôle de résidu})$$

où ε est une tolérance fixée. On a vu que pour la méthode de Newton le premier critère (contrôle de l'incrément) est optimal au sens qu'il garantit que l'erreur finale soit plus petite que la tolérance fixée.

Par contre, le deuxième critère (contrôle du résidu) est satisfaisant lorsque $|f'| \approx 1$ dans un voisinage de la racine α (un zéro pour la fonction f). Sinon il est soit trop restrictif (si $|f'| \gg 1$) soit trop faible (si $|f| \ll 1$).

Remarque

Dans le cas d'un système de deux équations non linéaires à 2 inconnues, par exemple :

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases} \quad (1.7)$$

L'algorithme de Newton-Raphson devient :

$$\begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \end{pmatrix} = \begin{pmatrix} x_1^k \\ x_2^k \end{pmatrix} - \begin{pmatrix} \frac{\partial f_1(x_1, x_2)}{\partial x_1} & \frac{\partial f_1(x_1, x_2)}{\partial x_2} \\ \frac{\partial f_2(x_1, x_2)}{\partial x_1} & \frac{\partial f_2(x_1, x_2)}{\partial x_2} \end{pmatrix}^{-1} \begin{pmatrix} f_1(x_1^k, x_2^k) \\ f_2(x_1^k, x_2^k) \end{pmatrix} \quad (1.8)$$

1.3 Equation Différentielles Ordinaires (EDO)

De nombreux modèles s'expriment au moyen d'équations différentielles ordinaires (en abrégé EDO). L'adjectif ordinaire est surtout là pour faire la distinction entre les EDO et les EDP (équations aux dérivées partielles). Si toutes les dérivées sont prises par rapport à une seule variable, on parle d'équation différentielle ordinaire (EDO). Une équation mettant en jeu des dérivées partielles est appelée équation aux dérivées partielles (EDP).

Une équation différentielle ordinaire, également notée EDO, d'ordre n est une relation entre la variable réelle $t \in I$, I intervalle réel une fonction inconnue $x(t)$ et ses dérivées par rapport à t définie par :

$$F(t, x, x'', \dots, x^{(n)}) = 0 \quad (1.9)$$

Une fonction x qui vérifie $F(t, x, x'', \dots, x^{(n)}) = 0$ s'appelle solution de l'EDO.

Remarque : Une EDO est d'ordre n si elle contient les dérivées de x jusqu'à l'ordre n .

Exemple1

Les équations suivantes :

$$\begin{aligned} y'(t) - t &= 0 \\ y''(t) - y(t) &= 0 \end{aligned}$$

Sont équation différentielles ordinaires.

Exemple2

L'EDO d'ordre 2 la plus célèbre est la deuxième **loi de Newton** :

$$m \frac{d^2 x}{dt^2} = F(x)$$

qui décrit par exemple la dynamique d'un point matériel soumis à la résultante des forces F .

On peut réécrire la loi de Newton sous la forme d'un système d'équations différentielles, en posant

$$v = \frac{dx}{dt}$$

On obtient alors le système du premier ordre autonome :

$$\begin{cases} v = \frac{dx}{dt} \\ \frac{dv}{dt} = \frac{1}{m} F(x) \end{cases}$$

1.4 Méthode de résolution :

1.4.1 Méthode d'Euler

La méthode d'Euler est une procédure numérique qui permet de résoudre de façon approximative des équations différentielles ordinaires du premier ordre avec condition initiale. Elle a le mérite d'être simple à comprendre et à programmer.

On cherche donc une solution approchée d'une équation ordinaire se mettant sous la forme

$$\begin{cases} \frac{dy}{dt} = f(t, y) & 0 \leq t \leq T \\ y(0) = y_0 \end{cases} \quad (1.10)$$

Avec T : une durée bien déterminée dans le temps.

On peut intégrer cette équation comme suit :

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t)) dt \quad (1.11)$$

La méthode d'Euler consiste à approcher l'intégrale par la méthode des rectangles à gauche :

$$\int_{t_k}^{t_{k+1}} f(t, y(t)) dt \cong h \times f(t_k, y(t_k))$$

D'où le schéma itératif suivant

$$y_{k+1} = y_k + \underbrace{(t_{k+1} - t_k)}_{h : \text{Taille de pas}} f(t_k, y_k)$$

Donc

$$\begin{cases} y_{k+1} = y_k + hf(t_k, y_k) \\ y_0 = y(0), \quad k = 0, 1, \dots, n \end{cases} \quad (1.12)$$

où y_k désigne l'approximation numérique de $y(t_k)$. La mise en œuvre est alors extrêmement simple :

Algorithme d'Euler

1. Initialisation du pas h et de la durée T .
2. Initialisation des conditions initiales : $t = 0$ et $y = y(0)$.
3. Tant que $t \leq T$ faire :
 - (a) Calcul de $k_1 = f(t, y)$.
 - (b) $y = y + hk_1$; $t = t + h$.
 - (c) Enregistrement des données.

Exemple1

Pour le problème de valeur initiale suivant:

$$\begin{cases} y' + 2y = 2 - e^{-4t} \\ y(0) = 1 \end{cases}$$

Utilisez la méthode d'Euler avec un pas de $h = 0,1$ pour trouver les valeurs approximatives de la solution à $t = 0.1, 0.2, 0.3, 0.4$ et 0.5 . Comparez-les aux valeurs exactes de la solution à ces points.

Il s'agit d'une équation différentielle linéaire assez simple, nous vous laisserons donc le soin de vérifier que la solution correcte est :

$$y(t) = 1 + \frac{1}{2}e^{-4t} - \frac{1}{2}e^{-2t}$$

De cela nous pouvons constater que :

$$f(t, y) = y' = 2 - e^{-4t} - 2y$$

Notez également que $t_0 = 0$ et $y_0 = y(0) = 1$.

Nous pouvons maintenant commencer à faire des calculs.

On a :

$$y_{k+1} = y_k + h \times f(t_k, y_k)$$

Pour $k = 0$

$$y_1 = y_0 + h \times f(t_0, y_0) \Rightarrow y_1 = 1 + 0.1f(0,1)$$

$$f_0 = f(0,1) = 2 - e^{-4(0)} - 2(1) = -1$$

$$\Rightarrow y_1 = 0.9$$

Donc, l'approximation de la solution à $t_1 = 0.1$ est $y_1 = 0.9$.

Pour $k = 1$

$$y_2 = y_1 + h \times f(t_1, y_1) \Rightarrow y_2 = 1 + 0.1f(0.1,0.9)$$

$$f_1 = f(0.1,0.9) = 2 - e^{-4(0.1)} - 2(0.9) = -0.4703$$

$$\Rightarrow y_2 = 0.9 + (0.1)(-0.4703) = 0.85297$$

Par conséquent, l'approximation de la solution à $t_2 = 0.2$ est $y_2 = 0.85297$.

Je vous laisse le soin de vérifier le reste de ces calculs.

$$f_2 = -0.1553$$

$$y_3 = 0.8374$$

$$f_3 = 0.02392$$

$$y_4 = 0.8398$$

$$f_4 = 0.11843$$

$$y_5 = 0.8517$$

Voici un tableau (1.1) rapide qui donne les approximations ainsi que la valeur exacte des solutions aux points donnés. Nous avons également inclus l'erreur en pourcentage. Il est souvent plus facile de voir à quel point une approximation est efficace si vous examinez les pourcentages. La formule pour cela est,

$$\text{Erreur (\%)} = \frac{|\text{valeur exacte} - \text{valeur approximée}|}{\text{valeur exacte}} \times 100$$

Tableau 1.1 Approximations et valeur exacte des solutions aux points donnés

Temps t_k	Approximation	Exacte	Erreur
$t_0 = 0$	$y_0 = 1$	$y(0) = 1$	0%
$t_1 = 0.1$	$y_1 = 0.9$	$y(0.1) = 0.9258$	2.79%
$t_2 = 0.2$	$y_2 = 0.8529$	$y(0.2) = 0.8895$	4.11%
$t_3 = 0.3$	$y_3 = 0.8374$	$y(0.3) = 0.8762$	4.42%
$t_4 = 0.4$	$y_4 = 0.8398$	$y(0.4) = 0.8763$	4.16%
$t_5 = 0.5$	$y_5 = 0.8517$	$y(0.5) = 0.8837$	3.63%

D'après le tableau (1.1), l'erreur maximale dans les approximations du dernier exemple était de 4,42%, ce qui n'est pas si mal, mais qui n'est pas non plus une si grande approximation. Ce type d'erreur est cependant généralement inacceptable dans la plupart des applications réelles. Alors, comment pouvons-nous obtenir de meilleures approximations?

Rappelons que nous obtenons les approximations en utilisant une ligne tangente pour approximer la valeur de la solution et que nous avançons dans le temps par pas de h . Donc, si nous voulons une approximation plus précise, il semble alors qu'il existe une seule façon d'obtenir une meilleure approximation c'est de ne pas avancer autant à chaque étape. En d'autres termes, il faudrait prendre des h plus petits.

Remarque

- La diminution de la taille de pas h améliore la précision de l'approximation.
- La diminution d'un facteur 10 la taille du pas h , réduit l'erreur d'un facteur de 10 environ.

1.4.2 Méthode de Runge-Kutta

Les techniques de *Runge-Kutta* sont des schémas numériques à un pas qui permettent de résoudre les équations différentielles ordinaires. Elles font parties des méthodes les plus populaires de par leur facilité de mise en œuvre et leur précision. C'est **Carle Runge** et **Martin Kutta** qui, au début du 20e siècle, ont inventé ces méthodes.

Dans de nombreux cas, les systèmes d'équations différentielles que l'on rencontre en science peuvent se mettre sous la forme d'une équation différentielle ordinaire du premier ordre (Eq. 1.8).

À l'instar de la *méthode d'Euler*, celles de *Runge-Kutta* sont des schémas numériques à un pas basée sur la discrétisation de la variable t . On note h ce pas et y_k la valeur approchée de $y(t_k)$ pour les différents instants $t_k = kh$.

En intégrant l'équation différentielle entre t_k et $t_k + h$ on a la relation :

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_k+h} f(t, y(t)) dt \quad (1.13)$$

L'idée consiste à approcher cette intégrale de façon plus précise que ne le fait la méthode d'Euler.

La méthode classique *de Runge-Kutta* est une approximation d'ordre 4.

On cherche donc une solution approchée d'une équation ordinaire se mettant sous la forme

$$\begin{cases} \frac{dy}{dt} = f(t, y) & 0 \leq t \leq T \\ t(0) = t_0, & y(0) = y_0 \end{cases}$$

Le schéma itératif de Runge-Kutta d'ordre 4 est (RK4):

$$\begin{cases} y_{k+1} = y_k + h \left[\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \right] \\ t_{k+1} = t_k + h \end{cases} \text{ avec } \begin{cases} k_1 = f(t_k, y_k) \\ k_2 = f(t_k + \frac{1}{2}h, y_k + \frac{1}{2}hk_1) \\ k_3 = f(t_k + \frac{1}{2}h, y_k + \frac{1}{2}hk_2) \\ k_4 = f(t_k + h, y_k + hk_3) \end{cases} \quad (1.14)$$

1.4.3 Méthode de d'Adams-Bashforth

La méthode d'Adams-Bashforth est une méthode à pas multiples qui repose sur une approximation de l'intégrale suivant :

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t)) dt$$

Les schémas itératifs de la méthode d'Adams-Bashforth:

On note en abrégé $f_k = f(t_k, y_k)$. Voici trois schémas :

- schéma explicite d'Adams-Bashforth d'ordre 2 à 2 pas (AB2):

$$y_{k+1} = y_k + \frac{h}{2} (3f_k - f_{k-1}) \quad (1.15)$$

- schéma explicite d'Adams-Bashforth d'ordre 3 à 3 pas (AB3) :

$$y_{k+1} = y_k + \frac{h}{12} (23f_k - 16f_{k-1} + 5f_{k-2}) \quad (1.16)$$

- schéma explicite d'Adams-Bashforth d'ordre 4 à 4 pas (AB4):

$$y_{k+1} = y_k + \frac{h}{24} (55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}) \quad (1.17)$$

1.4.4 Méthode d'Adams-Moulton (AM)

Cette méthode est l'une des méthodes fermées, son algorithme est :

$$y_{k+1} = y_k + \frac{h}{2} (f(t_k, y_k) + f(t_{k+1}, y_{k+1}^{(i)})) \quad (1.18)$$

Avec $y_{k+1}^{(i)}$ est donnée pour $i = 0, 1, 2, \dots$

$y_{k+1}^{(0)}$ est dite le prédicteur de y_{k+1} , les $y_{k+1}^{(i)}$ sont appelés les correcteurs, pour les calculés, on utilise la méthode d'Euler comme prédicteur et la méthode d'Adams-Moulton comme correcteur. On obtient :

$$\begin{cases} y_{k+1}^{(i)} = y_k + hf(t_k, y_k) & \text{Prédiction} \\ y_{k+1} = y_k + \frac{h}{2} (f(t_k, y_k) + f(t_{k+1}, y_{k+1}^{(i)})) & \text{Correction} \end{cases} \quad (1.19)$$