Chapitre 2

# Méthodes à pas multiples

## Introduction

Il existe une autre approche de résolution des équations différentielles qui a donné naissance à une famille de méthodes dites à pas multiples. L'idée de base de ces méthodes consiste à intégrer l'équation différentielle

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = \alpha \end{cases} \tag{1}$$

dans l'intervalle $[t_n,\ t_{n+1}]$ :

$\int_{t_n}^{t_{n+1}} y'(u)du = \int_{t_n}^{t_{n+1}} f(u,\ y(u))du$ d'où $y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(u,\ y(u))du$

Cela nous amène à un algorithme de la forme :

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(u,\ y(u))du \tag{2}$$

Il reste à trouver une approximation de l'intégrale dans le membre de droite. Pour y arriver, on utilise une interpolation de la fonction $f(u,\ y(u))$ à partir des valeurs de $y(u)$ calculées aux itérations précédentes. On note $f_n = f(t_n,\ y_n)$ l'approximation de $f(t_n,\ y(t_n))$ . Il est alors possible de construire une table de différences divisées pour cette fonction et d'effectuer l'interpolation par la méthode de Newton. Une première approche consiste à utiliser la table de différences divisées suivante.

**Rappel**

**Définition 1** *On définit les premières différences divisées de la fonction $f(t)$ par :*

$$f[t_n, t_{n+1}] = \frac{f(t_{n+1}) - f(t_n)}{t_{n+1} - t_n}$$

**Définition 2** *Les deuxièmes différences divisées de la fonction $f(t)$ sont définies à partir des premières différences divisées par la relation :*

$$f[t_i, t_{i+1}, t_{i+2}] = \frac{f[t_{i+1}, t_{i+2}] - f[t_i, t_{i+1}]}{t_{i+2} - t_i}$$

De même, les $n^{imes}$ différences divisées de la fonction $f(t)$ sont définies à partir des $(n-1)^{imes}$ différences divisées de la façon suivante :

$$f[t_0, t_1, t_2, \cdots t_n] = \frac{f[t_1, t_2, \cdots t_n] - f[t_0, t_1, t_2 \cdots t_{n-1}]}{t_n - t_0}$$

### 2.0.1 Première table des différences divisées

$$\begin{array}{llllll}
t_n & f_n & & & & \\
 & & f[t_n, t_{n-1}] & & & \\
t_{n-1} & f_{n-1} & & f[t_n, t_{n-1}, t_{n-2}] & & \\
 & & f[t_{n-1}, t_{n-2}] & & f[t_n, t_{n-1}, t_{n-2}, t_{n-3}] & \\
t_{n-2} & f_{n-2} & & f[t_{n-1}, t_{n-2}, t_{n-3}] & & \\
 & & f[t_{n-2}, t_{n-3}] & & & \\
t_{n-3} & f_{n-3} & & & &
\end{array}$$

On peut au besoin prolonger cette table. Le polynôme d'interpolation s'écrit :

$$\begin{aligned}
p_n(t) \;=\; & f_n + f[t_n,\ t_{n-1}](t - t_n) + f[t_n,\ t_{n-1},\ t_{n-2}](t - t_n)(t - t_{n-1}) \\
& + f[t_n,\ t_{n-1},\ t_{n-2},\ t_{n-3}](t - t_n)(t - t_{n-1})(t - t_{n-2}) + \cdots
\end{aligned}$$

On peut évaluer la fonction $f(t,\ y(t))$ au moyen de ce polynôme dans l'intervalle $[t_n,\ t_{n+1}]$. L'évaluation de $p(t)$ ne requiert que des valeurs connues provenant des pas de temps antérieurs. On peut aussi utiliser la table de différences divisées suivante.

### 2.0.2 Deuxième table de différences divisées

$$\begin{array}{llllll}
t_{n+1} & f_{n+1} & & & & \\
 & & f[t_n, t_{n+1}] & & & \\
t_n & f_n & & f[t_{n-1}, t_n, t_{n+1}] & & \\
 & & f[t_{n-1}, t_n] & & f[t_{n-2}, t_{n-1}, t_n, t_{n+1}] & \\
t_{n-1} & f_{n-1} & & f[t_{n-2}, t_{n-1}, t_n] & & \\
 & & f[t_{n-2}, t_{n-1}] & & & \\
t_{n-2} & f_{n-2} & & & &
\end{array}$$

Le polynôme correspondant est :

$$\begin{aligned}
p_n^*(t) = & f_{n+1} + f[t_n,\ t_{n+1}](t - t_{n+1}) + f[t_{n-1},\ t_n,\ t_{n+1}](t - t_{n+1})(t - t_n) \\
& + f[t_{n-2},\ t_{n-1},\ t_n,\ t_{n+1}](t - t_{n+1})(t - t_n)(t - t_{n-1}) + \cdots
\end{aligned}$$

On remarque immédiatement que l'évaluation de $p_n^*(t)$ requiert la connaissance préalable de $f_{n+1} = f(t_{n+1},\ y_{n+1})$ . Or, on ne connaît pas encore $y_{n+1}$. Nous verrons plus loin comment contourner cette difficulté.

Considérons d'abord le polynôme $p_n(t)$ . En augmentant successivement le degré du polynôme, on obtient des approximations de plus en plus précises que l'on peut insérer dans la relation (1). Par exemple, si l'on utilise le polynôme de degré 0, on a l'approximation $f(t,\ y(t)) \simeq p_0(t) = f_n$ et l'on trouve :

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f_n du = y_n + (t_{n+1} - t_n) f_n = y_n + h f(t_n,\ y(t_n)$$

qui n'est rien d'autre que l'expression de la méthode d'Euler explicite. En utilisant maintenant un polynôme de degré 1, on a l'approximation :

$$f(t,\ y(t)) \simeq p_1(t) = f_n + f[t_n,\ t_{n-1}](t - t_n)$$

En insérant cette expression dans l'équation (1), on obtient :

$$\begin{aligned} y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} (f_n + f[t_n,\ t_{n-1}](u - t_n)) du \\ &= y_n + (t_{n+1} - t_n) f_n + \frac{(f_n - f_{n-1})}{(t_n - t_{n-1})} \frac{(u - t_n)^2}{2} \Big|_{t_n}^{t_{n+1}} \\ &= y_n + \frac{h}{2}(3 f_n - f_{n-1}) \end{aligned}$$

ou encore :

$$y_{n+1} = y_n + \frac{h}{2}(3 f(t_n,\ y_n) - f(t_{n-1},\ y(t_{n-1})) \tag{3}$$

Dans l'équation précédente, on a posé $h = t_n - t_{n-1}$, ce qui suppose que le pas de temps est constant. On remarque qu'il s'agit d'une *méthode à deux pas*, en ce sens que pour obtenir $y_{n+1}$ on doit utiliser $y_n$ et $y_{n-1}$. Les méthodes vues jusqu'à maintenant (Euler, Taylor, Runge-Kutta, etc.) étaient à un pas.

Si on utilise un polynôme de degré 2, on a l'approximation :

$$f(t,\ y(t)) \simeq p_2(t) = f_n + f[t_n,\ t_{n-1}](t - t_n) + f[t_n,\ t_{n-1},\ t_{n-2}](t - t_n)(t - t_{n-1})$$

En insérant cette expression dans l'équation (1), on obtient :

$$y_{n+1} = y_n + \frac{h}{2}(3 f_n - f_{n-1}) + f[t_{n-2}, t_{n-1},\ t_n] \int_{t_n}^{t_{n+1}} (u - t_n)(u - t_{n-1}) du \tag{4}$$

1. On a $f[t_{n-2},\ t_{n-1},\ t_n] = \frac{f[t_{n-1}, t_n] - f[t_{n-2}, t_{n-1}]}{t_n - t_{n-2}} = \frac{f_n - 2 f_{n-1} + f_{n-2}}{2h}$
2. Pour le calcul de l'intégrale, on pose $\frac{(u - t_n)}{h} = s \Longleftrightarrow u - t_n = hs, u - t_{n-1} = u - t_n + t_n - t_{n-1} = hs + h,\ du = h ds$, ainsi $\int_{t_n}^{t_{n+1}} (u - t_n)(u - t_{n-1}) du = h^3 \int_0^1 s(s+1) ds = \frac{5}{6} h^3$

Ce qui donne

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}(3 f_n - f_{n-1}) + \frac{f_n - 2 f_{n-1} + f_{n-2}}{2h} \frac{5}{6} h^3 \\ &= y_n + \frac{h}{12}(23 f_n - 16 f_{n-1} + 5 f_{n-2}) \end{aligned}$$

On pourrait continuer ainsi en utilisant des polynômes de degré 3, 4, etc... En substituant ces polynômes dans l'équation (1), on obtient les formules d'Adams-Bashforth.

### 2.0.3 Formules d'Adams-Bashforth

$y_{n+1} = y_n + hf_n$ (ordre 1)

$y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1})$ (ordre 2)

$y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$ (ordre 3)

$y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$ (ordre 4)

**Remarque**

On définit l'erreur de troncature locale liée aux méthodes à pas multiples d'une manière semblable à ce que l'on fait dans le cas des méthodes à un pas.

**Définition 3** *Une méthode de résolution d'equations differentielles est dite à un pas si elle est de la forme*

$$y_{n+1} = y_n + h\phi\left(t_n, y_n\right)$$

*où $\phi$ est une fonction quelconque. Une telle relation est appelée équation aux différences. La méthode est à un pas si, pour obtenir la solution en $t = t_{n+1}$ ; on doit utiliser la solution numérique au temps $t_n$ seulement. On désigne méthodes à pas multiples les méthodes qui exigent également la solution numérique aux temps $t_{n-1}, t_{n-2}, t_{n-3} \cdots$.*

**Définition 4** *Lerreur de troncature locale au point $t = t_n$ est définie par :*

$$\tau_{n+1}(h) = \frac{y\left(t_{n+1}\right) - y\left(t_n\right)}{h} - \phi\left(t_n, y\left(t_n\right)\right)$$

**Exemple** le cas de la méthode d'Euler explicite ($\phi(t, y) = f(t, y)$ ). En effectuant un développement autour du point $t = t_n$, on trouve :

$$\begin{aligned} y\left(t_{n+1}\right) = y\left(t_n + h\right) &= y\left(t_n\right) + y'\left(t_n\right) h + \frac{y''\left(t_n\right) h^2}{2} + O\left(h^3\right) \\ &= y\left(t_n\right) + f\left(t_n, y\left(t_n\right)\right) h + \frac{y''\left(t_n\right) h^2}{2} + O\left(h^3\right) \end{aligned}$$

puisque $y'\left(t_n\right) = f\left(t_n, y\left(t_n\right)\right)$. L'erreur de troncature locale devient donc :

$$\tau_{n+1}(h) = \frac{y\left(t_{n+1}\right) - y\left(t_n\right)}{h} - f\left(t_n, y\left(t_n\right)\right) = \frac{y''\left(t_n\right) h}{2} + O\left(h^2\right)$$

ou plus simplement $\tau_{n+1}(h) = O(h)$ et la méthode d'Euler explicite converge donc à l'ordre $1 (p = 1)$. Dans ce qui suit, on indique sans démonstration l'ordre de l'erreur de troncature locale au fur et à mesure des besoins. On constate qu'en utilisant un polynôme de degré $n$ dans la relation (1) on obtient une méthode à $(n + 1)$ pas dont l'erreur de troncature locale est d'ordre $(n + 1)$.

Passons maintenant au polynôme $p_n^*(t)$ . On peut reprendre le raisonnement précédent. Si l'on utilise le polynôme de degré 0, on a l'approximation $f(t,\ y(t)) \simeq p_0^*(t) = f_{n+1}$ et l'on trouve :

$$f(t,\ y(t)) \simeq p_0^*(t)$$

et l'on trouve :

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f_{n+1} du = y_n + (t_{n+1} - t_n) f_{n+1} = y_n + h f(t_{n+1},\ y(t_{n+1}) \quad \text{(Euler implicite)}$$

En utilisant maintenant un polynôme de degré 1, on a l'approximation :

$$f(t,\ y(t)) \simeq p_1(t) = f_{n+1} + f[t_n,\ t_{n+1}](t - t_{n+1})$$

En insérant cette expression dans l'équation (1), on obtient :

$$\begin{aligned} y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} (f_{n+1} + f[t_n,\ t_{n+1}](u - t_{n+1}))du \\ &= y_n + (t_{n+1} - t_n) f_{n+1} + \frac{(f_{n+1} - f_n)}{(t_{n+1} - t_n)} \frac{(u - t_{n+1})^2}{2} \Big|_{t_n}^{t_{n+1}} \\ &= y_n + \frac{h}{2}(f_{n+1} + f_n) \end{aligned}$$

ou encore :

$$y_{n+1} = y_n + \frac{h}{2}(f(t_{n+1},\ y(t_{n+1})) + f(t_n,\ y(t_n)))$$

Si on utilise un polynôme de degré 2, on a l'approximation :

$$f(t,\ y(t)) \simeq p_2^*(t) = f_{n+1} + f[t_n,\ t_{n+1}](t - t_{n+1}) + f[t_{n-1},\ t_n,\ t_{n+1}](t - t_{n+1})(t - t_n)$$

En insérant cette expression dans l'équation (1), on obtient :

$$y_{n+1} = y_n + \frac{h}{2}(f_{n+1} + f_n) + f[t_{n-1}, t_n,\ t_{n+1}] \int_{t_n}^{t_{n+1}} (u - t_{n+1})(u - t_n)du \tag{5}$$

1. On a $f[t_{n-1},\ t_n,\ t_{n+1}] = \frac{f[t_{n+1}, t_n] - f[t_n, t_{n-1}]}{t_{n+1} - t_{n-1}} = \frac{f_{n+1} - 2f_n + f_{n-1}}{2h}$
2. Pour le calcule de l'intégrale, on pose $\frac{(u - t_{n+1})}{h} = s \iff u - t_{n+1} = hs, u - t_n = u - t_{n+1} + t_{n+1} - t_n = hs + h,\ du = hds$, ainsi $\int_{t_n}^{t_{n+1}} (u - t_n)(u - t_{n-1})du = h^3 \int_{-1}^{0} s(s+1)ds = \frac{-1}{6}h^3$

Ce qui donne

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}(f_{n+1} + f_n) + \frac{f_{n+1} - 2f_n + f_{n-1}}{2h} \frac{-1}{6} h^3 \\ &= y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1}) \end{aligned}$$

On peut ainsi passer à des polynômes de degré de plus en plus élevé dans l'équation $p_n^*(t)$ et obtenir les formules d'Adams-Moulton, qui sont résumées dans le tableau suivant.

### 2.0.4 Formules d'Adams-Moulton

$y_{n+1} = y_n + hf_{n+1}$ (ordre 1)

$y_{n+1} = y_n + \frac{h}{2}(f_{n+1} + f_n)$ (ordre 2)

$y_{n+1} = y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1})$ (ordre 3)

$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$ (ordre 4)

Les formules d'Adams-Moulton sont dites *implicites* en ce sens que les relations qui permettent d'évaluer $y_{n+1}$ dépendent de $y_{n+1}$ lui même.

Nous allons maintenant combiner les formules explicites d'Adams-Bashforth aux formules implicites d'Adams-Moulton en des schémas de type *prédicteurs- correcteurs*. Il s'agit simplement d'utiliser les schémas d'Adams-Bashforth pour obtenir une première approximation $y^p_{n+1}$ de $y_{n+1}$, qui est l'étape de prédiction. On fait appel ensuite aux formules d'Adams-Moulton pour corriger et éventuellement améliorer cette approximation. Il est important de remarquer que, dans ce cas, l'évaluation de $f_{n+1}$ dans les formules d'Adams-Moulton repose sur l'emploi de $y^p_{n+1}$, c'est-à-dire : $f_{n+1} \simeq f^p_{n+1} = f(t_{n+1},\ y^p_{n+1})$ On obtient ainsi les schémas suivants.

Schémas de prédiction-correction

| | |
|---|---|
| $y^p_{n+1} = y_n + hf_n$<br>$y_{n+1} = y_n + hf^p_{n+1}$ | ordre1 |
| $y^p_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1})$<br>$y_{n+1} = y_n + \frac{h}{2}(f^p_{n+1} + f_n)$ | ordre 2 |
| $y^p_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$<br>$y_{n+1} = y_n + \frac{h}{12}(5f^p_{n+1} + 8f_n - f_{n-1})$ | ordre 3 |
| $y^p_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$<br>$y_{n+1} = y_n + \frac{h}{24}(9f^p_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$ | ordre 4 |

**Remarque**

L'initialisation des méthodes de prédiction-correction nécessite l'usage d'une méthode à un pas. Si l'on prend par exemple le schéma d'ordre 4, il est clair que $n$ doit être plus grand ou égal à 3, car autrement on aurait besoin de $y_{-1}, y_{-2}$, etc. Or, au départ, seul $y_0$ est connu, provenant de la condition initiale. Les valeurs de $y_1$, de $y_2$ et de $y_3$ doivent être calculées à l'aide d'une autre méthode. Le plus souvent, on recourt à une méthode de Runge-Kutta qui est au moins du même ordre de convergence que la méthode de prédiction-correction que l'on souhaite utiliser.

**Exemple 1** *Soit l'équation différentielle :*

$$y'(t) = -y(t) + t + 1 \ (y(0) = 1)$$

On fait appel cette fois aux méthodes de prédiction-correction d'ordre 2 et 4. Les premiers pas de temps sont calculés par une méthode de Runge-Kutta d'ordre 4 qui a déjà servi à résoudre cette équation différentielle. La méthode

de prédiction-correction d'ordre 2 exige de connaître $y_0$, qui vaut 1, et $y_1$, qui a été calculé au préalable à l'aide de la méthode de Runge-Kutta d'ordre 4 et qui vaut 1,0048375 (une méthode de Runge-Kutta d'ordre 2 aurait été suffisante). La première itération donne d'abord une prédiction :

$$\begin{aligned} y_2^p &= y_1 + \frac{h}{2}(3f(t_1,\ y_1) - f(t_0,\ y_0)) \\ &= 1,0048375 + \frac{0,1}{2}(3f(0.1, 1.0048375) - f(0,1)) \\ &= 1.0048375 + 0.05(3(-1.0048375 + 0.1 + 1) - (-1 + 0 + 1)) \\ &= 1.019111875 \end{aligned}$$

et ensuite une correction :

$$\begin{aligned} y_2 = y_1 + \frac{h}{2}(f(t_2,\ y_2^p) + f(t_1,\ y_1)) \\ &= 1,0048375 + \frac{0,1}{2}(f(0.2, 1.019111875) + f(0.1, 1.0048375)) \\ &= 1.0048375 + 0,05(-1.019111875 + 02 + 1) + (-1,0048375 + 0.1 + 1)) \\ &= 1,\ 018640031 \end{aligned}$$

Les autres itérations sont résumées du tableau ci-dessous.

Schéma de prédiction-correction d'ordre 2

$$y'(t) = -y(t) + t + 1$$

| $t$ | $y_n$ | $y_n$ | $\lvert y(t_n) - y_n \rvert$ |
|---|---|---|---|
| $0,0$ | | $1,000000000$ | $0$ |
| $0,1$ | $1,019111875$ | $1,004837500$ | $0,819\ 640 \times 10^{-7}$ |
| $0,2$ | $1,041085901$ | $1,018640031$ | $0,907\ 218 \times 10^{-4}$ |
| $0,3$ | $1,\ 070487675$ | $1,040653734$ | $0,164\ 486 \times 10^{-3}$ |
| $0,4$ | $1,106614851$ | $1,070096664$ | $0,223\ 381 \times 10^{-3}$ |
| $0,5$ | $1,148826758$ | $1,106261088$ | $0,269\ 571 \times 10^{-3}$ |
| $0,6$ | $1,\ 196543746$ | $1,148506695$ | $0,304\ 940 \times 10^{-3}$ |
| $0,7$ | $1,249241382$ | $1,196254173$ | $0,331129 \times 10^{-3}$ |
| $0,8$ | $1,\ 306445195$ | $1,\ 248979396$ | $0,349\ 568 \times 10^{-3}$ |
| $0,9$ | $1,367725911$ | $1306208166$ | $0,361493 \times 10^{-3}$ |
| $1,0$ | | $1,367511462$ | $0,367979 \times 10^{-3}$ |

L'erreur à $t = 0,1$ est beaucoup plus faible qu'aux autres valeurs de $t$ puisque la solution numérique à cet endroit a été calculée à l'aide d'une méthode d'ordre 4. Cela explique également l'absence de prédicteur pour cette valeur.

De manière similaire, la méthode de prédiction-correction d'ordre 4 requiert le calcul de $y_1$, de $y_2$ et de $y_3$ à l'aide de la méthode de Runge-Kutta d'ordre 4. Par la suite, il suffit d'utiliser l'algorithme :

$$\begin{aligned} y_{n+1}^p &= y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f) \\ y_{n+1} &= y_n + \frac{h}{24}(9f_{n+1}^p + 19f_n - 5f_{n-1} + f_{n-2}) \end{aligned}$$

pour $n \geq 3$. La première itération s'effectue comme suit :

$$\begin{aligned} y_4^p = y_3 + \frac{0,1}{24}&(55f(t_3,\ y_3) - 59f(t_2,\ y_2) + 37f(t_1,\ y_1) - 9f(t_0,\ y)) \\ &= 1,0408184 + 0,1(55f(\mathrm{O},3,1,0408184) - 59f(0,2,1,0187309) \end{aligned}$$

$$+37f(0,1,1,0048375) - 9f(0,1)) = 1,1070323$$

$$\begin{aligned} y_4 = y_3 + \frac{0,1}{24}&(9f(t_4,\ y_4^p) + 19f(t_3,\ y_3) - 5f(t_2,\ y_2) + f(t_1,\ y)) \\ &= 1,0408184 + 0,1(9f(0,4,1,1070323) + 19f(0,3,1,0408184) \end{aligned}$$

$$-5f(0,2,1,0187309) + f(0,1,1,0048375)) = 1,0703199$$

On obtient enfin les résultats du tableau ci-dessous.

Schéma de prédiction-correction d'ordre 4

$$y'(t) = -y(t) + t + 1$$

| $t$ | $y_n$ | $y_n$ | $\lvert y(t_n) - y_n \rvert$ |
|---|---|---|---|
| $0,0$ | | $1,0000000$ | $0$ |
| $0,1$ | | $1,\ 0048375$ | $0,819\ 640 \times 10^{-7}$ |
| $0,2$ | $1,1070323$ | $1,0187309$ | $0,148\ 328 \times 10^{-6}$ |
| $0,3$ | $1,1065332$ | $1,0408184$ | $0,201319 \times 10^{-6}$ |
| $0,4$ | $1,1488136$ | $1,\ 0703199$ | $0,127\ 791 \times 10^{-6}$ |
| $0,5$ | $1,\ 1965869$ | $1,1065303$ | $0,391302 \times 10^{-6}$ |
| $0,6$ | $1,2493302$ | $1,1488110$ | $0,603\ 539 \times 10^{-6}$ |
| $0,7$ | $1,3065706$ | $1,\ 1965845$ | $0,772415 \times 10^{-6}$ |
| $0,8$ | $1,3678801$ | $1,2493281$ | $0,903\ 669 \times 10^{-6}$ |
| $0,9$ | | $1,3065687$ | $0,100\ 294 \times 10^{-5}$ |
| $1,0$ | | $1,\ 3678784$ | $0,107\ 514 \times 10^{-5}$ |

L'erreur est ici beaucoup plus petite qu'avec la méthode d'ordre 2.

**Remarque**

La précision des méthodes de prédiction-correction est comparable à celle des méthodes à un pas d'ordre équivalent. Si l'on considère, par exemple, le schéma d'ordre 4, les résultats sont comparables à ceux de la méthode de Runge-Kutta d'ordre 4. Par contre, si l'on calcule le coût lié à l'emploi d'une méthode selon le nombre d'évaluations de la fonction $f(t,\ y)$ à chaque itération, on se rend compte que la méthode de prédiction-correction d'ordre 4 est moins coûteuse. En effet, chaque itération de la méthode de Runge-Kutta nécessite 4 évaluations de la fonction $f(t,\ y)$ , alors que la méthode de prédiction-correction n'exige que 2 nouvelles évaluations, compte tenu de celles qui ont déjà été effectuées aux itérations précédentes. Cela est également vrai dans le cas des méthodes d'ordre 2 et 3