In [1]:

```
from IPython.display import display, Latex
from IPython.core.display import HTML
css_file = './custom.css'
HTML(open(css_file, "r").read())
```

Out[1]:

M62_CM6-7 : Schémas multistep

Table of Contents

1 Construction de schémas

Considérons le problème de Cauchy

> trouver une fonction y: $I \subset \mathbb{R} \to \mathbb{R}$ définie sur un intervalle $I = [t_0, T]$ telle que
> $$\begin{cases} y'(t) = \varphi(t, y(t)), & \forall t \in I = [t_0, T], \\ y(t_0) = y_0, \end{cases}$$
> avec y_0 une valeur donnée et supposons que l'on ait montré l'existence et l'unicité d'une solution y pour $t \in I$.

Pour $h > 0$ soit $t_n \overset{\text{déf.}}{=} t_0 + nh$ avec $n = 0, 1, 2, \dots, N_h$ une suite de $N_h + 1$ nœuds de I induisant une discrétisation de I en N_h sous-intervalles $I_n = [t_n; t_{n+1}]$ chacun de longueur $h > 0$ (appelé le *pas de discrétisation*).

Pour chaque nœud t_n, on cherche la valeur inconnue u_n qui approche la valeur exacte $y_n \equiv y(t_n)$.

- L'ensemble de $N_h + 1$ valeurs $\{t_0, t_1 = t_0 + h, \dots, t_{N_h} = T\}$ représente les points de la discrétisation.
- L'ensemble de $N_h + 1$ valeurs $\{y_0, y_1, \dots, y_{N_h}\}$ représente la solution exacte.
- L'ensemble de $N_h + 1$ valeurs $\{u_0 = y_0, u_1, \dots, u_{N_h}\}$ représente la solution numérique. Cette solution approchée sera obtenue en construisant une suite récurrente.

Les schémas qu'on va construire permettent de calculer (explicitement ou implicitement) u_{n+1} à partir de $u_n, u_{n-1}, \dots, u_{n-k}$ et il est donc possible de calculer successivement u_1, u_2,..., en partant de u_0 par une formule de récurrence de la forme

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = \Phi(u_{n+1}, u_n, u_{n-1}, \dots, u_{n-k}), & \forall n \in \mathbb{N}. \end{cases}$$

Les schémas qu'on va construire permettent ainsi de calculer (explicitement ou implicitement) u_{n+1} à partir de $u_n, u_{n-1}, \dots, u_{n-k}$ et il est donc possible de calculer successivement u_1, u_2,\dots, en partant de

u_0.

Méthodes explicites et méthodes implicites
Une méthode est dite *explicite* si la valeur u_{n+1} peut être calculée directement à l'aide des valeurs précédentes u_k, $k \leq n$ (ou d'une partie d'entre elles).
Une méthode est dite *implicite* si u_{n+1} n'est défini que par une relation implicite faisant intervenir la fonction φ.

Méthodes à un pas et méthodes multi-pas
Une méthode numérique pour l'approximation du problème de Cauchy est dite *à un pas* si pour tout $n \in \mathbb{N}$, u_{n+1} ne dépend que de u_n et éventuellement de lui-même.
Autrement, on dit que le schéma est une méthode *multi-pas* (ou à pas multiples).

Dans la suite nous allons écrire explicitement ces schémas. Pour vérifier nos calculs d'interpolation puis intégration, nous allons utiliser le package de calcul formel `SymPy` .

In [2]:

```
import sympy as symb
symb.init_printing()
symb.var('h,t,t_n')
symb.var('phi_np1,phi_n,phi_nm1,phi_nm2,phi_nm3,phi_nm4,phi_nm
symb.var('  y_np1,  y_n,  y_nm1,  y_nm2,  y_nm3,  y_nm4,  y_nm
t_np1=t_n+h
t_nm1=t_n-h
t_nm2=t_n-2*h
t_nm3=t_n-3*h
t_nm4=t_n-4*h
t_nm5=t_n-5*h
```

1.1 Schémas d'Adam

Si nous intégrons l'EDO $y'(t) = \varphi(t, y(t))$ entre t_n et t_{n+1} nous obtenons

$$y_{n+1} - y_n = \int_{t_n}^{t_{n+1}} \varphi(t, y(t))dt.$$

On peut construire différentes schémas selon la formule de quadrature utilisée pour approcher le membre de droite. Cette solution approchée sera obtenue en construisant une suite récurrente comme suit:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} \text{un polynôme d'interpolation de } \varphi(t, u)\mathrm{d}t. \end{cases}$$

Les **schémas d'Adam** approchent l'intégrale $\int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale d'un polynôme p **interpolant** φ **en des points donnés qui peuvent être à l'extérieur de l'intervalle d'intégration** $[t_n\,; t_{n+1}]$. On

peut construire différentes schémas selon les points d'interpolation choisis. Ils se divisent en deux familles: les méthodes d'*Adam-Bashforth* qui sont explicites et les méthodes d'*Adam-Moulton* qui sont implicites:

- **schémas AB-q:** les schémas d'Adam-Bashforth d'ordre q approchent l'intégrale $\int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale $\int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t$ où p est le polynôme interpolant φ en
$$\{t_n, t_{n-1}, t_{n+1-q}\} \text{ avec } q \geq 1;$$
- **schémas AM-q:** les schémas d'Adam-Moulton d'ordre q approchent l'intégrale $\int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale $\int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t$ où p est le polynôme interpolant φ en
$$\{t_{n+1}, t_n, t_{n-1}, t_{n+1-q}\} \text{ avec } q \geq 0.$$

Notons que, pour calculer successivement u_q, u_{q+1}, ..., il faut d'abord initialiser $u_0, u_1, \ldots, u_{q-1}$. Cette initialisation doit être faite par des approximations adéquates car seul u_0 est donné.

Remarque: la condition de A-stabilité est de plus en plus contraignante quand l'ordre augmente.

1.1.1 AB-1

On a

- Points d'interpolation: $\{(t_n, \varphi(t_n, y_n))\}$
- Polynôme: $p(t) = \varphi(t_n, y(t_n))$
- Approximation: $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = h\varphi(t_n, y(t_n))$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} = u_n + h\varphi(t_n, u_n) \quad n = 0, 1, \ldots N-1 \end{cases}$$
La méthode AB_1 coïncide avec la méthode d'Euler progressive.

In [3]:

```
p=symb.interpolate([(t_n,phi_n)], t)
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[3]:

$h\phi_n$

1.1.2 AB-2

On a

- Points d'interpolation: $\{(t_n, \varphi(t_n, y_n)); (t_{n-1}, \varphi(t_{n-1}, y_{n-1}))\}$
- Polynôme:
$p(t) = \varphi(t_n, y_n)\frac{t-t_{n-1}}{t_n-t_{n-1}} + \varphi(t_{n-1}, y(t_{n-1}))\frac{t-t_n}{t_{n-1}-t_n} = \varphi(t_n, y_n)\frac{t-t_{n-1}}{h} +$

- Approximation:
  $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{2}(3\varphi(t_n, y(t_n)) - \varphi(t_{n-1}, y(t_{n-1})))$et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_{n+1} = u_n + \frac{h}{2}(3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})) \quad n = 1, 2, \dots N-1 \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction AB_1.

In [4]:

```
p=symb.interpolate([(t_n,phi_n),(t_nm1,phi_nm1)], t)
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[4]:

$$\frac{h}{2}(3\phi_n - \phi_{nm1})$$

1.1.3 AB-3

On a

- Points d'interpolation:
  $\{(t_n, \varphi(t_n, y_n)); (t_{n-1}, \varphi(t_{n-1}, y_{n-1})); (t_{n-1}, \varphi(t_{n-2}, y_{n-2}))\}$
- Polynôme:
  $$\begin{aligned} p(t) &= \varphi(t_n, y_n)\frac{(t-t_{n-1})(t-t_{n-2})}{(t_n-t_{n-1})(t_n-t_{n-2})} + \varphi(t_{n-1}, y(t_{n-1}) \\ &\quad + \varphi(t_{n-2}, y(t_{n-2}))\frac{(t-t_n)(t-t} \\ &= \frac{\varphi(t_{n-2}, y_{n-2})}{2h^2}(t-t_{n-1})(t-t_n) - \frac{\varphi(t_{n-1}, y_{n-1})}{h^2}(t-t_{n-2})(t \end{aligned}$$
- Approximation:
  $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{12}(23\varphi(t_n, y(t_n)) - 16\varphi(t_{n-1}, y(t_{n-1})) \cdot$
  et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_2 = u_1 + \frac{h}{2}(3\varphi(t_1, u_1) - \varphi(t_0, u_0)) \approx y(t_2) \\ u_{n+1} = u_n + \frac{h}{12}(23\varphi(t_n, u_n) - 16\varphi(t_{n-1}, u_{n-1} + 5\varphi(t_{n-2}, u_{n-2})) \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction AB_1 et u_2 est une approximation de $y(t_2)$ obtenue en utilisant la méthode AB_2.

In [5]:

```
p=symb.interpolate([(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi_nm2
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[5]:

$$\frac{h}{12}(23\phi_n - 16\phi_{nm1} + 5\phi_{nm2})$$

1.1.4 AB-4

On a

- Points d'interpolation:
  $\{(t_n, \varphi(t_n, y_n)); (t_{n-1}, \varphi(t_{n-1}, y_{n-1})); (t_{n-1}, \varphi(t_{n-2}, y_{n-2}))\}$
- Polynôme: $p(t) = \sum_{i=n-2}^{n} \left(\varphi(t_i, y_i) \prod_{\substack{j=n-2,\dots,n \\ j\neq i}} \frac{t - t_j}{t_i - t_j} \right)$
- Approximation:
  $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{24}(55\varphi(t_n, y(t_n)) - 59\varphi(t_{n-1}, y(t_{n-1}))$ ·
  et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_2 = u_1 + \frac{h}{2}(3\varphi(t_1, u_1) - \varphi(t_0, u_0)) \approx y(t_2) \\ u_3 = u_2 + \frac{h}{12}(23\varphi(t_2, u_2) - 16\varphi(t_1, u_1 + 5\varphi(t_0, u_0)) \approx y(t_3) \\ u_{n+1} = u_n + \frac{h}{24}(55\varphi(t_n, u_n) - 59\varphi(t_{n-1}, u_{n-1} + 37\varphi(t_{n-2}, u_{n-2}) \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction AB_1, u_2 est une approximation de $y(t_2)$ obtenue en utilisant la méthode AB_2 etc...

In [6]:

```
p=symb.interpolate([(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi_nm2
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[6]:

$$\frac{h}{24}(55\phi_n - 59\phi_{nm1} + 37\phi_{nm2} - 9\phi_{nm3})$$

1.1.5 AM-0

On a

- Points d'interpolation: $\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1}))\}$
- Polynôme: $p(t) = \varphi(t_{n+1}, y(t_{n+1}))$
- Approximation: $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = h\varphi(t_{n+1}, y(t_{n+1}))$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} = u_n + h\varphi(t_{n+1}, u_{n+1}) \quad n = 0, 1, \dots N-1 \end{cases}$$
La méthode AM_0 coïncide avec la méthode d'Euler regressive.

In [7]:

```
p=symb.interpolate([(t_np1,phi_np1)], t)
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[7]:

$h\phi_{np1}$

1.1.6 AM-1

On a

- Points d'interpolation: $\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1})), (t_n, \varphi(t_n, y_n))\}$
- Polynôme: $p(t) = \varphi(t_{n+1}, y_{n+1})\frac{t-t_n}{t_{n+1}-t_n} + \varphi(t_n, y_n)\frac{t-t_{n+1}}{t_n-t_{n+1}}$
- Approximation:
$y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{2}(\varphi(t_{n+1}, y_{n+1}) + \varphi(t_n, y_n))$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} = u_n + \frac{h}{2}(\varphi(t_n, u_n) + \varphi(t_{n+1}, u_{n+1})) \quad n = 0, 1, \dots N-1 \end{cases}$$
La méthode AM_1 coïncide avec la méthode de Crank-Nicolson.

In [8]:

```
p=symb.interpolate([(t_np1,phi_np1),(t_n,phi_n)], t)
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[8]:

$\frac{h}{2}\left(\phi_n + \phi_{np1}\right)$

1.1.7 AM-2

On a

- Points d'interpolation:
$\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1})), (t_n, \varphi(t_n, y_n)), (t_{n-1}, \varphi(t_{n-1}, y_{n-1}))\}$

- Polynôme:
  $p(t) = \varphi(t_{n+1}, y_{n+1})\frac{(t-t_n)(t-t_{n-1})}{(t_{n+1}-t_n)(t_{n+1}-t_{n-1})} + \varphi(t_n, y_n)\frac{(t-t_{n+1})(t-t_{n-1})}{(t_n-t_{n+1})(t_n-t_{n-1})} + \varphi($
- Approximation:
  $y_{n+1} - y_n \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{12}(5\varphi(t_{n+1}, y_{n+1}) + 8\varphi(t_n, y_n) - \varphi(t_{n-}$
  et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + \frac{h}{2}(\varphi(t_1, u_1) + \varphi(t_0, u_0)) \\ u_{n+1} = u_n + \frac{h}{12}(5\varphi(t_{n+1}, u_{n+1}) + 8\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1}) \quad n \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction AM_1.

In [9]:

```
p=symb.interpolate([(t_np1,phi_np1),(t_n,phi_n),(t_nm1,phi_nm1
symb.integrate(p,(t,t_n,t_np1)).simplify()
```

Out[9]:

$$\frac{h}{12}\left(8\phi_n - \phi_{nm1} + 5\phi_{np1}\right)$$

1.1.8 Calcul systématique des coefficients des méthodes AB et AM

In [10]:

```
import sympy as symb
symb.init_printing()
symb.var('h,t,t_n')
symb.var('phi_np1,phi_n,phi_nm1,phi_nm2,phi_nm3,phi_nm4,phi_nm
symb.var('  y_np1,  y_n,  y_nm1,  y_nm2,  y_nm3,  y_nm4,  y_nm
t_np1=t_n+h
t_nm1=t_n-h
t_nm2=t_n-2*h
t_nm3=t_n-3*h
t_nm4=t_n-4*h
t_nm5=t_n-5*h

Points=[(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi_nm2),(t_nm3,phi
for q in range(1,len(Points)):
    p=symb.interpolate(Points[0:q+1], t)
    AB=(symb.integrate(p,(t,t_n,t_np1)).simplify())
    print("AB-",q)
    display(AB)
    print("\n")

print("\n")

Points=[(t_np1,phi_np1),(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi
for q in range(len(Points)):
    p=symb.interpolate(Points[0:q+1], t)
    AM=(symb.integrate(p,(t,t_n,t_np1)).simplify())
    print("AM-",q)
    display(AM)
    print("\n")
```

AB- 1

$$\frac{h}{2}(3\phi_n - \phi_{nm1})$$

AB- 2

$$\frac{h}{12}(23\phi_n - 16\phi_{nm1} + 5\phi_{nm2})$$

AB- 3

$$\frac{h}{24}(55\phi_n - 59\phi_{nm1} + 37\phi_{nm2} - 9\phi_{nm3})$$

AB- 4

$$\frac{h}{720}(1901\phi_n - 2774\phi_{nm1} + 2616\phi_{nm2} - 1274\phi_{nm3} + 251$$

AB- 5

$$\frac{h}{1440}(4277\phi_n - 7923\phi_{nm1} + 9982\phi_{nm2} - 7298\phi_{nm3} + 28$$

AM- 0

$$h\phi_{np1}$$

AM- 1

$$\frac{h}{2}\left(\phi_n + \phi_{np1}\right)$$

AM- 2

$$\frac{h}{12}\left(8\phi_n - \phi_{nm1} + 5\phi_{np1}\right)$$

AM- 3

$$\frac{h}{24}\left(19\phi_n - 5\phi_{nm1} + \phi_{nm2} + 9\phi_{np1}\right)$$

AM- 4

$$\frac{h}{720}\left(646\phi_n - 264\phi_{nm1} + 106\phi_{nm2} - 19\phi_{nm3} + 251\phi_{np1}\right)$$

AM- 5

$$\frac{h}{1440}\left(1427\phi_n - 798\phi_{nm1} + 482\phi_{nm2} - 173\phi_{nm3} + 27\phi_{nm}\right.$$

AM- 6

$$\frac{h}{60480}\left(65112\phi_n - 46461\phi_{nm1} + 37504\phi_{nm2} - 20211\phi_{nm}\right.$$

1.2 Schémas de Nyström et de Milne-Simpson

Les méthodes d'Adam peuvent être facilement généralisées en intégrant l'EDO $y'(t) = \varphi(t, y(t))$ entre t_{n-r} et t_{n+1} avec $r \geq 1$.

Avec $r = 1$, si nous intégrons l'EDO $y'(t) = \varphi(t, y(t))$ entre t_{n-1} et t_{n+1} nous obtenons

$$y_{n+1} - y_{n-1} = \int_{t_{n-1}}^{t_{n+1}} \varphi(t, y(t))dt.$$

On peut construire différentes schémas selon la formule de quadrature utilisée pour approcher le membre de droite. Cette solution approchée sera obtenue en construisant une suite récurrente comme suit:

$$\begin{cases} u_0 = y_0, \\ u_1 = ?, \\ u_{n+1} = u_{n-1} + \int_{t_{n-1}}^{t_{n+1}} \text{un polynôme d'interpolation de } \varphi(t, u)\mathrm{d}t. \end{cases}$$

On obtient les schémas de Nyström, qui sont explicites, et les schémas de Milne-Simpson, qui sont implicites:

- **schémas N-q**: les schémas de Nyström d'ordre q approchent l'intégrale $\int_{t_{n-1}}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale $\int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t$ où p est le polynôme interpolant φ en
$$\{t_n, t_{n-1}, t_{n+1-q}\} \text{ avec } q \geq 1;$$
- **schémas MS-q**: les schémas de Milne-Simpson d'ordre q approchent l'intégrale $\int_{t_{n-1}}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale $\int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t$ où p est le polynôme interpolant φ en
$$\{t_{n+1}, t_n, t_{n-1}, t_{n+1-q}\} \text{ avec } q \geq 0.$$

1.2.1 N-1

On a

- Points d'interpolation: $\{(t_n, \varphi(t_n, y_n))\}$
- Polynôme: $p(t) = \varphi(t_n, y(t_n))$
- Approximation: $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = 2h\varphi(t_n, y(t_n))$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_{n+1} = u_{n-1} + 2h\varphi(t_n, u_n) \qquad n = 1, 2, \dots N-1 \end{cases}$$
où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction d'Euler explicite.
La méthode N_1 coïncide avec la méthode du point milieu (appelée aussi *Saute-mouton* ou *Leapfrog*).

In [11]:

```
p=symb.interpolate([(t_n,phi_n)], t).simplify()
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[11]:

$2h\phi_n$

1.2.2 N-2

On a

- Points d'interpolation: $\{(t_n, \varphi(t_n, y_n)), (t_{n-1}, \varphi(t_{n-1}, y_{n-1}))\}$
- Polynôme: $p(t) = \varphi(t_n, y(t_n))\frac{t-t_{n-1}}{t_n-t_{n-1}} + \varphi(t_{n-1}, y_{n-1})\frac{t-t_n}{t_{n-1}-t_n}$
- Approximation: $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = 2h\varphi(t_n, y(t_n))$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_{n+1} = u_{n-1} + 2h\varphi(t_n, u_n) \qquad n = 1, 2, \dots N-1 \end{cases}$$
où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction d'Euler explicite.
On retrouve la méthode N_1.

In [12]:

```
p=symb.interpolate([(t_n,phi_n),(t_nm1,phi_nm1)], t).simplify(
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[12]:

$2h\phi_n$

1.2.3 N-3

On a

- Points d'interpolation:
$\{(t_n, \varphi(t_n, y_n)); (t_{n-1}, \varphi(t_{n-1}, y_{n-1})); (t_{n-1}, \varphi(t_{n-2}, y_{n-2}))\}$
- Polynôme:
$$\begin{aligned} p(t) &= \varphi(t_n, y_n)\frac{(t-t_{n-1})(t-t_{n-2})}{(t_n-t_{n-1})(t_n-t_{n-2})} + \varphi(t_{n-1}, y_{n-1}) \\ &\quad + \varphi(t_{n-2}, y_{n-2})\frac{(t-t_n)(t-t_n} \\ &= \frac{\varphi(t_{n-2}, y_{n-2})}{2h^2}(t-t_{n-1})(t-t_n) - \frac{\varphi(t_{n-1}, y_{n-1})}{h^2}(t-t_{n-2})(t \end{aligned}$$

- Approximation:
  $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{3}(7\varphi(t_n, y_n) - 2\varphi(t_{n-1}, y_{n-1}) + \varphi(t_n$
  et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_2 = u_0 + 2h\varphi(t_1, u_1) \approx y(t_2) \\ u_{n+1} = u_{n-1} + \frac{h}{3}(7\varphi(t_n, u_n) - 2\varphi(t_{n-1}, u_{n-1} + \varphi(t_{n-2}, u_{n-2})) \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction d'Euler explicite et u_2 est une approximation de $y(t_2)$ obtenue en utilisant la méthode N_2.

In [13]:

```
p=symb.interpolate([(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi_nm2
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[13]:

$$\frac{h}{3}(7\phi_n - 2\phi_{nm1} + \phi_{nm2})$$

1.2.4 MS-0

On a

- Points d'interpolation: $\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1}))\}$
- Polynôme: $p(t) = \varphi(t_{n+1}, y(t_{n+1}))$
- Approximation: $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = 2h\varphi(t_{n+1}, y(t_{n+1}))$ et on obtient le schéma
  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_{n+1} = u_{n-1} + 2h\varphi(t_{n+1}, u_{n+1}) \quad n = 1, 2, \dots N-1 \end{cases}$$
  où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction d'Euler explicite.

In [14]:

```
p=symb.interpolate([(t_np1,phi_np1)], t).simplify()
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[14]:

$$2h\phi_{np1}$$

1.2.5 MS-1

On a

- Points d'interpolation: $\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1})), (t_n, \varphi(t_n, y_n))\}$
- Polynôme: $p(t) = \varphi(t_{n+1}, y_{n+1})\frac{t-t_n}{t_{n+1}-t_n} + \varphi(t_n, y_n)\frac{t-t_{n+1}}{t_n-t_{n+1}}$
- Approximation: $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = 2h\varphi(t_n, y_n)$ et on obtient le schéma

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \approx y(t_1) \\ u_{n+1} = u_{n-1} + 2h\varphi(t_n, u_n) \qquad n = 1, 2, \dots N-1 \end{cases}$$

où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction d'Euler explicite.

In [15]:

```
p=symb.interpolate([(t_np1,phi_np1),(t_n,phi_n)], t).simplify(
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[15]:

$2h\phi_n$

1.2.6 MS-2

On a

- Points d'interpolation:
  $\{(t_{n+1}, \varphi(t_{n+1}, y_{n+1})), (t_n, \varphi(t_n, y_n)), (t_{n-1}, \varphi(t_{n-1}, y_{n-1}))\}$
- Polynôme:
  $p(t) = \varphi(t_{n+1}, y_{n+1})\frac{(t-t_n)(t-t_{n-1})}{(t_{n+1}-t_n)(t_{n+1}-t_{n-1})} + \varphi(t_n, y_n)\frac{(t-t_{n+1})(t-t_{n-1})}{(t_n-t_{n+1})(t_n-t_{n-1})} + \varphi($
- Approximation:
  $y_{n+1} - y_{n-1} \approx \int_{t_{n-1}}^{t_{n+1}} p(t)\mathrm{d}t = \frac{h}{3}(\varphi(t_{n+1}, y_{n+1}) + 4\varphi(t_n, y_n) + \varphi(t_{n-}$
  et on obtient le schéma

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + \frac{h}{2}(\varphi(t_1, u_1) + \varphi(t_0, u_0)) \\ u_{n+1} = u_{n-1} + \frac{h}{3}(\varphi(t_{n+1}, u_{n+1}) + 4\varphi(t_n, u_n) + \varphi(t_{n-1}, u_{n-1}) \qquad n \end{cases}$$

où u_1 est une approximation de $y(t_1)$ obtenue en utilisant une prédiction AM_1.

In [16]:

```
p=symb.interpolate([(t_np1,phi_np1),(t_n,phi_n),(t_nm1,phi_nm1
symb.integrate(p,(t,t_nm1,t_np1)).simplify()
```

Out[16]:

$$\frac{h}{3}\left(4\phi_n + \phi_{nm1} + \phi_{np1}\right)$$

1.2.7 Calcul systématique des coefficients des méthodes N et MS

In [17]:

```
import sympy as symb
symb.init_printing()
symb.var('h,t,t_n')
symb.var('phi_np1,phi_n,phi_nm1,phi_nm2,phi_nm3,phi_nm4,phi_nm
symb.var('  y_np1,  y_n,  y_nm1,  y_nm2,  y_nm3,  y_nm4,  y_nm
t_np1=t_n+h
t_nm1=t_n-h
t_nm2=t_n-2*h
t_nm3=t_n-3*h
t_nm4=t_n-4*h
t_nm5=t_n-5*h

Points=[(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi_nm2),(t_nm3,phi
for q in range(1,len(Points)):
    p=symb.interpolate(Points[0:q+1], t)
    N=(symb.integrate(p,(t,t_nm1,t_np1)).simplify())
    print("N-",q)
    display(N)
    print("\n")

print("\n")

Points=[(t_np1,phi_np1),(t_n,phi_n),(t_nm1,phi_nm1),(t_nm2,phi
for q in range(len(Points)):
    p=symb.interpolate(Points[0:q+1], t)
    MS=(symb.integrate(p,(t,t_nm1,t_np1)).simplify())
    print("MS-",q)
    display(MS)
    print("\n")
```

N- 1

$$2h\phi_n$$

N- 2

$$\frac{h}{3}(7\phi_n - 2\phi_{nm1} + \phi_{nm2})$$

N- 3

$$\frac{h}{3}(8\phi_n - 5\phi_{nm1} + 4\phi_{nm2} - \phi_{nm3})$$

N- 4

$$\frac{h}{90}(269\phi_n - 266\phi_{nm1} + 294\phi_{nm2} - 146\phi_{nm3} + 29\phi_{nm4})$$

N- 5

$$\frac{h}{90}(297\phi_n - 406\phi_{nm1} + 574\phi_{nm2} - 426\phi_{nm3} + 169\phi_{nm4} -$$

MS- 0

$$2h\phi_{np1}$$

MS- 1

$$2h\phi_n$$

MS- 2

$$\frac{h}{3}\left(4\phi_n + \phi_{nm1} + \phi_{np1}\right)$$

MS- 3

$$\frac{h}{3}\left(4\phi_n + \phi_{nm1} + \phi_{np1}\right)$$

MS- 4

$$\frac{h}{90}\left(124\phi_n + 24\phi_{nm1} + 4\phi_{nm2} - \phi_{nm3} + 29\phi_{np1}\right)$$

MS- 5

$$\frac{h}{90}\left(129\phi_n + 14\phi_{nm1} + 14\phi_{nm2} - 6\phi_{nm3} + \phi_{nm4} + 28\phi_{np1}\right)$$

MS- 6

$$\frac{h}{3780}\left(5640\phi_n + 33\phi_{nm1} + 1328\phi_{nm2} - 807\phi_{nm3} + 264\phi_{n}\right.$$

1.3 Schémas BDF (Backward Differentiation

Formulae)

Si nous interpolons la fonction inconnue $t \mapsto y(t)$ par un polynôme p en des points donnés

$$\{t_{n+1}, t_n, \dots, t_{n+1-i}\} \text{ avec } q \geq 1,$$

nous obtenons $y(t) \simeq p(t)$ et nous pouvons écrire

$$\varphi(t, y) = y'(t) \simeq p'(t).$$

En évaluant cette relation en t_{n+1} nous obtenons la relation

$$\varphi(t_{n+1}, y_{n+1}) \simeq p'(t_{n+1}).$$

On peut construire différents schémas (implicites) selon les points d'interpolation utilisés pour approcher la fonction inconnue $t \mapsto y(t)$. Cette solution approchée sera obtenue en construisant une suite récurrente comme suit:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} \text{ solution de l'équation } \varphi(t_{n+1}, u_{n+1}) \simeq p'(t_{n+1}). \end{cases}$$

Remarque: la condition de A-stabilité est de plus en plus contraignante quand l'ordre augmente.

1.3.1 BDF-1

On a

- Points d'interpolation: $\{(t_{n+1}, y_{n+1}), (t_n, y_n)\}$
- Polynôme: $p(t) = \frac{y_{n+1}-y_n}{h}(t - t_n) + y_n$
- Dérivée: $p'(t) = \frac{y_{n+1}-y_n}{h}$
- Approximation: $\varphi(t_{n+1}, y_{n+1}) \approx p'(t_{n+1}) = \frac{y_{n+1}-y_n}{h}$ et on obtient le schéma

  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} \text{ solution de l'équation } \varphi(t_{n+1}, u_{n+1}) = \frac{u_{n+1}-u_n}{h} \quad n = 1, 2, \end{cases}$$

  c'est-à-dire le schéma

  $$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} = u_n + h\varphi(t_{n+1}, u_{n+1}) \quad n = 0, 1, \dots N-1. \end{cases}$$

  La méthode BDF_1 coïncide avec la méthode d'Euler implicite.

In [18]:

```
p=symb.interpolate([(t_np1,y_np1),(t_n,y_n)], t).factor()
print("p(t)")
display(p)
dp=symb.diff(p,t).subs(t,t_np1)
print("p'(t)")
display(dp)
print("y_np1=")
symb.solve(dp-phi_np1,y_np1)[0]
```

p(t)

$$-\frac{1}{h}\left(-hy_n + ty_n - ty_{np1} - t_n y_n + t_n y_{np1}\right)$$

p'(t)

$$-\frac{1}{h}\left(y_n - y_{np1}\right)$$

y_np1=

Out[18]:

$$h\phi_{np1} + y_n$$

1.3.2 BDF-2

On a

- Points d'interpolation: $\{(t_{n+1}, y_{n+1}), (t_n, y_n), (t_{n-1}, y_{n-1})\}$
- Polynôme:
$$p(t) = y_{n+1}\frac{(t-t_n)(t-t_{n-1})}{(t_{n+1}-t_n)(t_{n+1}-t_{n-1})} + y_n\frac{(t-t_{n+1})(t-t_{n-1})}{(t_n-t_{n+1})(t_n-t_{n-1})} +$$
$$= \frac{(t-t_n)(t-t_{n-1})}{2h^2}y_{n+1} + \frac{(t-t_{n+1})(t-t_{n-1})}{-h^2}y_n + \frac{(t-t_{n+1}}{-2}$$
- Dérivée:
$$p'(t) = \frac{(t-t_n)+(t-t_{n-1})}{2h^2}y_{n+1} + \frac{(t-t_{n+1})+(t-t_{n-1})}{-h^2}y_n + \frac{(t-t_{n+1})+(t-t_n)}{-2h^2}y_{n-1}$$
- Approximation:
$\varphi(t_{n+1}, y_{n+1}) \approx p'(t_{n+1}) = \frac{3}{2h}y_{n+1} - \frac{2}{h}y_n + \frac{1}{2h}y_{n-1}$ et on obtient le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_{n+1} \text{ solution de l'équation } \varphi(t_{n+1}, u_{n+1}) = \frac{3}{2h}u_{n+1} - \frac{2}{h}u_n + \frac{1}{2h} \end{cases}$$
c'est-à-dire le schéma
$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_1, u_1) \\ u_{n+1} = \frac{4}{3}u_n - \frac{1}{3}u_{n-1} + \frac{2}{3}h\varphi(t_{n+1}, u_{n+1}) \quad n = 1, 2, 3, \ldots N-1 \end{cases}$$

In [19]:

```
p=symb.interpolate([(t_np1,y_np1),(t_n,y_n),(t_nm1,y_nm1)], t)
print("p(t)")
display(p)
dp=symb.diff(p,t).subs(t,t_np1)
print("p'(t)")
display(dp)
print("y_np1=")
symb.solve(dp-phi_np1,y_np1)[0]
```

p(t)

$$\frac{1}{h^2}\left(h^2 y_n + \frac{h}{2}\left(-t y_{nm1} + t y_{np1} + t_n y_{nm1} - t_n y_{np1}\right) - t^2 y_n + \frac{t^2}{} \right.$$

$$\left. + \frac{t_n^2 y_n}{2}\right.$$

p'(t)

$$\frac{1}{h^2}\left(\frac{h}{2}\left(-y_{nm1} + y_{np1}\right) + 2 t_n y_n - t_n y_{nm1} - t_n y_{np1} - 2 y_n \left(h + t_n\right.\right.$$

y_np1=

Out[19]:

$$\frac{2h}{3}\phi_{np1} + \frac{4y_n}{3} - \frac{y_{nm1}}{3}$$

1.3.3 Calcul systématique des coefficients des méthodes BDF

In [20]:

```
import sympy as symb
symb.init_printing()
symb.var('h,t,t_n')
symb.var('phi_np1')
symb.var('  u_np1,  u_n,  u_nm1,  u_nm2,  u_nm3,  u_nm4,  u_nm
t_np1=t_n+h
t_nm1=t_n-h
t_nm2=t_n-2*h
t_nm3=t_n-3*h
t_nm4=t_n-4*h
t_nm5=t_n-5*h

Points=[(t_np1,u_np1),(t_n,u_n),(t_nm1,u_nm1),(t_nm2,u_nm2),(t
for q in range(1,len(Points)):
    print("BDF-",q)
    p=symb.interpolate(Points[0:q+1], t)
    dp=symb.diff(p,t).subs(t,t_np1)
    sol=symb.solve(dp-phi_np1,u_np1)
    display(sol)
    print("\n")
```

BDF- 1

$$\left[h\phi_{np1} + u_n\right]$$

BDF- 2

$$\left[\frac{2h}{3}\phi_{np1} + \frac{4u_n}{3} - \frac{u_{nm1}}{3}\right]$$

BDF- 3

$$\left[\frac{6h}{11}\phi_{np1} + \frac{18u_n}{11} - \frac{9u_{nm1}}{11} + \frac{2u_{nm2}}{11}\right]$$

BDF- 4

$$\left[\frac{12h}{25}\phi_{np1} + \frac{48u_n}{25} - \frac{36u_{nm1}}{25} + \frac{16u_{nm2}}{25} - \frac{3u_{nm3}}{25}\right]$$

BDF- 5

$$\left[\frac{60h}{137}\phi_{np1} + \frac{300u_n}{137} - \frac{300u_{nm1}}{137} + \frac{200u_{nm2}}{137} - \frac{75u_{nm3}}{137} + \frac{12u}{1}\right.$$

BDF- 6

$$\left[\frac{20h}{49}\phi_{np1} + \frac{120u_n}{49} - \frac{150u_{nm1}}{49} + \frac{400u_{nm2}}{147} - \frac{75u_{nm3}}{49} + \frac{24\iota}{4}\right.$$

1.4 Schémas multi-pas de type predictor-corrector

Lorsqu'on utilise une méthode implicite, pour calculer u_{n+1} on doit résoudre une équation (en générale non-linéaire), par exemple avec une méthode de point fixe.
Une approche différente qui permet de s'affranchir de cette étape est donnée par les méthodes **predictor-corrector**. Une méthode predictor-corrector est une méthode qui permet de calculer u_{n+1} de façon explicite à partir d'une méthode implicite comme suit.

On considère une méthode implicite $u_{n+1} = u_n + hG(u_{n+1}; u_n, \dots)$:

- étape de **prédiction**: on calcule $\tilde{u}_{n+1}$ une approximation de u_{n+1} par une méthode explicite;
- étape de **correction**: on "corrige" la méthode implicite en approchant $G(u_{n+1}; u_n, \dots)$ par $G(\tilde{u}_{n+1})$.

Remarque
Ces méthodes héritent de l'ordre de précision du correcteur. Cependant, étant explicites, elles sont soumises à une condition de stabilité qui est typiquement celle du prédicteur. Elles ne sont donc pas adaptées à la résolution des problèmes de Cauchy sur des intervalles non bornés ou des problèmes stiff.

1.4.1 Exemple : schéma de Heun

La méthode de Heun

$$\begin{cases} u_0 = y_0 \\ \tilde{u}_{n+1} = u_n + h\varphi(t_n, u_n) & n = 1, 2, \dots N-1 \\ u_{n+1} = u_n + \frac{h}{2}(\varphi(t_n, u_n) + \varphi(t_{n+1}, \tilde{u}_{n+1})) & n = 1, 2, \dots N-1 \end{cases}$$

est construite par les deux étapes suivantes:

- predictor: méthode d'Euler explicite (ou AB_1) $\tilde{u}_{n+1} = u_n + h\varphi(t_n, u_n)$
- corrector: méthode de Crank-Nicolson (ou AM_1) $u_{n+1} = u_n + \frac{h}{2}(\varphi(t_n, u_n) + \varphi(t_{n+1}, u_{n+1}))$

1.4.2 Exemple : AB-AM

Des méthodes de type predictor-corrector sont souvent construites en utilisant une prédiction d'Adam-Bashforth suivie d'une correction d'Adam-Moulton.
Par exemple, si on considère les deux étapes suivantes

- predictor: méthode AB_2 $\tilde{u}_{n+1} = u_n + \frac{h}{2}(3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1}))$
- corrector: méthode AM_2
  $u_{n+1} = u_n + \frac{h}{12}(5\varphi(t_{n+1}, u_{n+1}) + 8\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})$on obtient la méthode AB_2-AM_2:
  $$\begin{cases} u_0 = y_0 \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ \tilde{u}_{n+1} = u_n + \frac{3}{2}h\left(\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})\right) & n \\ u_{n+1} = u_n + \frac{h}{12}(5\varphi(t_{n+1}, \tilde{u}_{n+1}) + 8\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1}) & n \end{cases}$$

2 Exercice : Interpolation, Quadrature et EDO

- Soit f une fonction de classe $\mathscr{C}1([-1, 1])$. Écrire le polynôme $p \in \mathbb{R}_2[\tau]$ qui interpole f aux points -1, 0 et 1.
- Construire une méthode de quadrature comme suit:
  $$\int_0^1 f(\tau)\mathrm{d}\tau \approx \int_0^1 p(\tau)\mathrm{d}\tau.$$
  NB: on intègre sur $[0, 1]$ mais on interpole en -1, 0 et 1.
- À l'aide d'un changement de variable affine entre l'intervalle $[0, 1]$ et l'intervalle $[a, b]$, en déduire une formule de quadrature pour l'intégrale
  $$\int_a^b f(x)\mathrm{d}x$$
  lorsque f est une fonction de classe $\mathscr{C}1([2a - b, b])$.
  Remarque: $[2a - b, b] = [a - (b - a), a + (b - a)]$
- Considérons le problème de Cauchy: trouver y: $[t_0, T] \subset \mathbb{R} \to \mathbb{R}$ tel que
  $$\begin{cases} y'(t) = \varphi(t, y(t)), \quad \forall t \in [t_0, T], \\ y(t_0) = y_0, \end{cases}$$
  dont on suppose l'existence d'une unique solution y.

  On subdivise l'intervalle $[t_0\,; T]$ en N intervalles $[t_n\,; t_{n+1}]$ de largeur $h = \dfrac{T - t_0}{N}$ avec $t_n = t_0 + nh$ pour $n = 0, \ldots, N$. Utiliser la formule obtenue au point 3 pour approcher l'intégrale
  $$\int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t.$$
  En déduire un schéma à deux pas implicite pour l'approximation de la solution du problème de Cauchy.

2.1 Correction

On cherche les coefficients α, β et γ du polynôme $p(\tau) = \alpha + \beta\tau + \gamma\tau^2$ tels que

$$\begin{cases} p(-1) = f(-1), \\ p(0) = f(0), \\ p(1) = f(1), \end{cases} \quad \text{c'est à dire} \quad \begin{cases} \alpha - \beta + \gamma = f(-1), \\ \alpha = f(0), \\ \alpha + \beta + \gamma = f(1). \end{cases}$$

Donc $\alpha = f(0)$, $\beta = \frac{f(1)-f(-1)}{2}$ et $\gamma = \frac{f(1)-2f(0)+f(-1)}{2}$.

In [21]:

```
import sympy as symb
symb.init_printing()

symb.var('f_0,f_1,f_m1,')

p=symb.interpolate([(1,f_1),(0,f_0),(-1,f_m1)], t).factor(t)
p
```

Out[21]:

$$-\frac{1}{2}\left(-2f_0 + t^2\left(2f_0 - f_1 - f_{m1}\right) + t\left(-f_1 + f_{m1}\right)\right)$$

2.2 Correction

On en déduit la méthode de quadrature

$$\int_0^1 f(\tau)\mathrm{d}\tau \approx \int_0^1 p(\tau)\mathrm{d}\tau = \alpha + \frac{\beta}{2} + \frac{\gamma}{3} = \frac{-f(-1) + 8f(0) + 5f(1)}{12}.$$

In [22]:

```
q=(symb.integrate(p,(t,0,1)).simplify())
q
```

Out[22]:

$$\frac{2f_0}{3} + \frac{5f_1}{12} - \frac{f_{m1}}{12}$$

2.3 Correction

Soit $x = m\tau + q$, alors

$$\int_a^b f(x)\mathrm{d}x = m\int_0^1 f(m\tau + q)\mathrm{d}\tau \quad \text{avec} \quad \begin{cases} a = q, \\ b = m + q, \end{cases} \quad \backslash\text{ie} \quad \{$$

d'où le changement de variable $x = (b-a)\tau + a$. On en déduit la formule de quadrature

$$\int_a^b f(x)\mathrm{d}x = (b-a)\int_0^1 f\left((b-a)\tau + a\right)\mathrm{d}\tau \approx (b-a)\frac{-f(2a-b)+8}{12}$$

2.4 Correction

On pose $a = t_n$ et $b = t_{n+1}$ d'où la formule de quadrature

$$\int_{t_n}^{t_{n+1}} f(t)\mathrm{d}t \approx (t_{n+1}-t_n)\frac{-f(2t_n - t_{n+1}) + 8f(t_n) + 5f(t_{n+1})}{12} = h\frac{-f(t_{n-}}{}$$

En utilisant la formule de quadrature pour l'intégration de l'EDO $y'(t) = \varphi(t, y(t))$ entre t_n et t_{n+1} on obtient

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t \approx h\frac{-\varphi(t_{n-1}, y(t_{n-1})) + 8\varphi(t_n, y(t_n}{12}$$

Si on note u_n une approximation de la solution y au temps t_n, on obtient le schéma à deux pas implicite suivant:

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 \text{ à définir}, \\ u_{n+1} = u_n + h\dfrac{-\varphi(t_{n-1}, u_{n-1}) + 8\varphi(t_n, u_n) + 5\varphi(t_{n+1}, u_{n+1})}{12} \quad n = \end{cases}$$

On peut utiliser une pédiction d'Euler explicite pour initialiser u_1:

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_{n+1} = u_n + h\dfrac{-\varphi(t_{n-1}, u_{n-1}) + 8\varphi(t_n, u_n) + 5\varphi(t_{n+1}, u_{n+1})}{12} \quad n = \end{cases}$$

3 Exercice : Interpolation, Quadrature et EDO

1. Soit $h > 0$ et $f\colon [a-h, a+h] \to \mathbb{R}$ une fonction de classe $\mathscr{C}^1([a-h, a+h])$. Écrire le polynôme $p \in \mathbb{R}_1[x]$ qui interpole f aux points $a-h$ et a, i.e. l'équation de la droite $p \in \mathbb{R}_1[x]$ qui passe par les deux points $(a-h, f(a-h))$ et $(a, f(a))$.
2. Construire une méthode de quadrature comme suit:
$$\int_a^{a+h} f(x)\mathrm{d}x \approx \int_a^{a+h} p(x)\mathrm{d}x.$$
NB: on intègre sur $[a, a+h]$ mais on interpole en $a-h$ et a.
3. Considérons le problème de Cauchy: trouver $y\colon [t_0, T] \subset \mathbb{R} \to \mathbb{R}$ tel que
$$\begin{cases} y'(t) = \varphi(t, y(t)), \quad \forall t \in [t_0, T], \\ y(t_0) = y_0, \end{cases}$$
dont on suppose l'existence d'une unique solution y.
On subdivise l'intervalle $[t_0; T]$ en N intervalles $[t_n; t_{n+1}]$ de largeur

$h = \dfrac{T - t_0}{N}$ avec $t_n = t_0 + nh$ pour $n = 0, \dots, N$.

Utiliser la formule obtenue au point 2 pour approcher l'intégrale

$$\int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t.$$

En déduire un schéma à deux pas explicite pour l'approximation de la solution du problème de Cauchy.

3.1 Correction

$$p(x) = \frac{f(a) - f(a-h)}{a - (a-h)}(x-a) + f(a) = \frac{f(a) - f(a-h)}{h}(x-a) + f($$

In [23]:

```
import sympy as symb
symb.init_printing()

symb.var('a,h,f_a,f_amh')

p=symb.interpolate([(a,f_a),(a-h,f_amh)], t).factor(t)
p
```

Out[23]:

$$\frac{1}{h}(-af_a + af_{amh} + f_a h + t(f_a - f_{amh}))$$

3.2 Correction

On en déduit la méthode de quadrature

$$\begin{aligned}
\int_a^{a+h} f(x)\mathrm{d}x &\approx \int_a^{a+h} p(x)\mathrm{d}x \\
&= \frac{f(a) - f(a-h)}{h}\left[\frac{(x-a)^2}{2}\right]_a^{a+h} + f(a)[x]_a^{a+h} \\
&= \frac{f(a) - f(a-h)}{2h}\left((a+h-a)^2 - (a-a)^2\right) + f(a)(a - \\
&= \frac{f(a) - f(a-h)}{2h}h^2 + hf(a) \\
&= h\frac{3f(a) - f(a-h)}{2}.
\end{aligned}$$

In [24]:

```
q=(symb.integrate(p,(t,a,a+h)).simplify())
q
```

Out[24]:

$$\frac{h}{2}(3f_a - f_{amh})$$

3.3 Correction

On pose $a = t_n$ et $a + h = t_{n+1}$ d'où la formule de quadrature

$$\int_{t_n}^{t_{n+1}} f(t)\mathrm{d}t \approx (t_{n+1} - t_n)\frac{3f(t_n) - f(2t_n - t_{n+1})}{2} = h\frac{3f(t_n) - f(t_{n-1})}{2}.$$

En utilisant la formule de quadrature pour l'intégration de l'EDO $y'(t) = \varphi(t, y(t))$ entre t_n et t_{n+1} on obtient

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t \approx h\frac{3\varphi(t_n, y(t_n)) - \varphi(t_{n-1}, y(t_{n-1}))}{2}$$

Si on note u_n une approximation de la solution y au temps t_n, on obtient le schéma à deux pas implicite suivant:

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 \text{ à définir}, \\ u_{n+1} = u_n + h\dfrac{3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})}{2} \qquad n = 1, 2, \dots N-1 \end{cases}$$

On peut utiliser une prédiction d'Euler explicite pour initialiser u_1:

$$\begin{cases} u_0 = y(t_0) = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_{n+1} = u_n + h\dfrac{3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})}{2} \qquad n = 1, 2, \dots N-1 \end{cases}$$

4 Exercice: Construction d'un schéma

Considérons le problème de Cauchy

> trouver une fonction y: $I \subset \mathbb{R} \to \mathbb{R}$ définie sur un intervalle $I = [t_0, T]$ telle que
>
> $$\begin{cases} y'(t) = \varphi(t, y(t)), & \forall t \in I = [t_0, T], \\ y(t_0) = y_0, \end{cases}$$
>
> avec y_0 une valeur donnée et supposons que l'on ait montré l'existence et l'unicité d'une solution y pour $t \in I$.

On subdivise l'intervalle $I = [t_0; T]$, avec $T < +\infty$, en N intervalles $[t_n; t_{n+1}]$ de largeur $h = (T - t_0)/N$ avec $t_n = t_0 + nh$ pour $n = 0, \dots, N$.
Pour chaque nœud t_n, on note $y_n = y(t_n)$ et on cherche la valeur inconnue u_n qui approche la valeur exacte y_n

Si nous intégrons l'EDO $y'(t) = \varphi(t, y(t))$ entre t_{n-2} et t_{n+1} nous obtenons

$$y_{n+1} - y_{n-2} = \int_{t_{n-2}}^{t_{n+1}} \varphi(t, y(t))dt.$$

Écrire le schéma obtenu en approchant l'intégrale $\int_{t_{n-2}}^{t_{n+1}} \varphi(t, y(t))\mathrm{d}t$ par l'intégrale $\int_{t_{n-2}}^{t_{n+1}} p(t)\mathrm{d}t$ où p est le polynôme interpolant φ en t_n et t_{n-1} (attention à bien initialiser la suite définie par récurrence pour qu'on puisse effectivement calculer tous les termes). Le schéma obtenu est-il explicite?

4.1 Correction

Le polynôme interpolant φ en t_n et t_{n-1} a équation

$$p(t) = \frac{\varphi(t_n, y_n) - \varphi(t_{n-1}, y_{n-1})}{t_n - t_{n-1}}(t - t_n) + \varphi(t_n, y_n).$$

On intègre ce polynôme entre t_{n-2} et t_{n+1} :

$$\begin{aligned}\int_{t_{n-2}}^{t_{n+1}} p(t)\mathrm{d}t &= \frac{\varphi(t_n, y_n) - \varphi(t_{n-1}, y_{n-1})}{h}\left[\frac{(t - t_n)^2}{2}\right]_{t_{n-2}}^{t_{n+1}} + \varphi(t_n, y_n)[t]_{t_{n-2}}^{t_{n+1}} \\ &= \frac{\varphi(t_n, y_n) - \varphi(t_{n-1}, y_{n-1})}{2h}\left[(t_{n+1} - t_n)^2 - (t_{n-2} - t_n)^2\right] \cdot \\ &= \frac{3}{2}h\left(\varphi(t_n, y_n) + \varphi(t_{n-1}, y_{n-1})\right).\end{aligned}$$

On obtient le schéma explicite

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_{n+1} = u_{n-2} + \frac{3}{2}h\left(\varphi(t_n, u_n) + \varphi(t_{n-1}, u_{n-1})\right). \end{cases}$$

5 Exercice : Déduction d'un schéma Predictor-Corrector

Écrire le schéma predictor-corrector basé sur AM-3 AB-2. Attention à bien initialiser la suite récurrente.

5.1 Correction

- AB-1 $u_{n+1} = u_n + h\varphi(t_n, u_n)$
- AB-2 $u_{n+1} = u_n + \frac{h}{2}(3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1}))$

- AB-3
  $u_{n+1} = u_n + \frac{h}{12}(23\varphi(t_n, u_n) - 16\varphi(t_{n-1}, u_{n-1}) + 5\varphi(t_{n-2}, u_{n-2}))$
- AM-3
  $u_{n+1} = u_n + \frac{h}{24}(9\varphi(t_{n+1}, u_{n+1}) + 19\varphi(t_n, u_n) - 5\varphi(t_{n-1}, u_{n-1}) + q$
  donc
  $$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0) \\ u_2 = u_1 + \frac{h}{2}(3\varphi(t_1, u_1) - \varphi(t_0, u_0)) \\ \tilde{u} = u_n + \frac{h}{12}(23\varphi(t_n, u_n) - 16\varphi(t_{n-1}, u_{n-1}) + 5\varphi(t_{n-2}, u_{n-2})) \\ u_{n+1} = u_n + \frac{h}{24}(9\varphi(t_{n+1}, \tilde{u}) + 19\varphi(t_n, u_n) - 5\varphi(t_{n-1}, u_{n-1}) + \varphi(\end{cases}$$

6 Exercice : Ordre d'un schéma

Une méthode numérique à un pas a été utilisée pour résoudre une équation différentielle avec condition initiale. Les résultats obtenus par cette méthode en prenant des pas de temps $h = 0.1$, $h = 0.05$ et $h = 0.025$ sont donnés dans le tableau suivant (remarque: une valeur sur deux est affichée pour $h = 0.05$ et une valeur sur quatre est affichée pour $h = 0.025$)

t_i	y_i pour $h = 0.1$	y_i pour $h = 0.05$	y_i pour $h = 0.025$
1.0	0.500000	0.500000	0.500000
1.1	0.512084	0.512242	0.512280
1.2	0.511698	0.512101	0.512196
1.3	0.500927	0.501559	0.501704
1.4	0.482686	0.483447	0.483619
1.5	0.459861	0.460633	0.460804

En calculant le rapport des erreurs et sachant que $y(1.5) = 0.460857$, déterminer l'ordre de la méthode numérique utilisée.

6.1 Correction

La méthode utilisée est d'ordre 2, en effet, pour $t = 1.5$ nous avons

$$\frac{E(h = 0.05)}{E(h = 0.025)} = \frac{|0.460633 - 0.460857|}{|0.460804 - 0.460857|} = 4.226 \simeq 2^2.$$

Si on a accès à `polyfit` on peut tracer la courbe d'erreur en echèlle logaritmique et estimer la pente:

In [25]:

```
%reset -f
%matplotlib inline
%autosave 300

from matplotlib.pylab import *

H=[0.1,0.05,0.025]
err=[]
y=0.460857
err.append(abs(y-0.459861))
err.append(abs(y-0.460633))
err.append(abs(y-0.460804))

print ('Ordre\t %1.2f' %(polyfit(log(H),log(err), 1)[0]))

subplot(1,2,1)
loglog(H,err, 'r-o');
grid()
subplot(1,2,2)
plot(log(H),log(err), 'r-o');
```

```
Autosaving every 300 seconds
Ordre    2.12
```