Sup'Galilée
MACS1

Année 2021/2022

# Analyse numérique - TD1 - Corrigé
## Algorithmique

Convention algorithmique (voir le cours) :
- la $i$-ème composante d'un tableau 1D ou vecteur $v$ sera notée $v(i)$, et le premier indice est $i = 1$.
- la composante $(i, j)$ d'un tableau 2D ou matrice $A$ sera notée $A(i, j)$, et les indices commencent à $i = 1$ et $j = 1$.

### Exercice 1

1. Ecrire la fonction SP permettant de calculer : $y = \sum_{i=1}^{m} a_i \prod_{j=1}^{n} b_j \sin\left(\frac{2j\pi}{n} x^i\right)$.
2. On veut calculer
$$I = \prod_{k=0}^{n} \left[\alpha_k \sum_{i=0}^{p} \cos\left(\frac{2\pi}{k+i+1} z\right) + \beta_k \sum_{i=0}^{q} \prod_{\substack{j=0 \\ j\neq i}}^{q} \frac{z - x_j}{x_i - x_j}\right].$$
   (a) Quelles sont les données minimales permettant de calculer $I$ ?
   (b) Ecrire en langage algorithmique la fonction CalculI permettant de calculer $I$.

Correction

1. Ici pour le problème soit bien posé, il faut préciser que les données sont : $\boldsymbol{a} \in \mathbb{R}^m$, $\boldsymbol{b} \in \mathbb{R}^n$, $x \in \mathbb{R}$, et que le résultat est $y \in \mathbb{R}$, avec $y = \sum_{i=1}^{m} a_i \prod_{j=1}^{n} b_j \sin\left(\frac{2j\pi}{n} x^i\right)$.

**Remarque 1.** *Notons que la dimension du vecteur $\boldsymbol{a}$, c'est-à-dire $m$, peut-être traitée de deux façons :*

*(a) mise en paramètre d'entrée de la fonction SP ; dans ce cas toutes les données sont explicites mais un inconvénient est que l'utilisateur de la fonction pourrait entrer une mauvaise valeur pour $m$ ... l'algorithme devrait renvoyer un message d'erreur dans ce cas, à l'utilisateur.*

*(b) mise de façon implicite, dans le paramètre d'entrée $\boldsymbol{a}$ ; en effet, dans la plupart des langages (MATLAB, Octave, Python,...), lorsque l'on manipule des tableaux/vecteurs il est possible de récupérer leur dimension (même en C, si on utilise une structure, la dimension étant alors un champ de cette structure). Cela permet de minimiser les paramètres d'entrée des fonctions. Néanmoins il peut y avoir des problèmes (fortran par ex. ou langage C si $\boldsymbol{a}$ est de type double *).*

*Dans la suite on considère l'option (b) : on suppose que l'on dispose d'une fonction `length`, permettant de retourner la taille $m$ d'un vecteur $\boldsymbol{a} \in \mathbb{R}^m$ : $m \leftarrow$ `length`$(\boldsymbol{a})$.*

**Algorithm 1** Fonction SP : Etape 1

**Données :** $\boldsymbol{a}$ : vecteur de $\mathbb{R}^m$
$\boldsymbol{b}$ : vecteur de $\mathbb{R}^n$
$x$ : réel
**Résultat :** $y$ : réel

1: **Fonction** $y \leftarrow \mathrm{SP}(\boldsymbol{a}, \boldsymbol{b}, x)$
2: $m \leftarrow$ length$(\boldsymbol{a})$
3: $n \leftarrow$ length$(\boldsymbol{b})$
4: $y \leftarrow 0$
5: **Pour** $i \leftarrow 1$ **à** $m$ **faire**
6: $p \leftarrow \prod_{j=1}^{n} b_j \sin((2j\pi/n)x^i)$
7: $y \leftarrow y + a(i) * p$
8: **fin Pour**
9: **fin Fonction**

$\Longrightarrow$

**Algorithm 2** Fonction SP : Etape 2

**Données :** $\boldsymbol{a}$ : vecteur de $\mathbb{R}^m$
$\boldsymbol{b}$ : vecteur de $\mathbb{R}^n$
$x$ : réel
**Résultat :** $y$ : réel

1: **Fonction** $y \leftarrow \mathrm{SP}(\boldsymbol{a}, \boldsymbol{b}, x)$
2: $m \leftarrow$ length$(\boldsymbol{a})$
3: $n \leftarrow$ length$(\boldsymbol{b})$
4: $y \leftarrow 0$
5: **Pour** $i \leftarrow 1$ **à** $m$ **faire**
6: $t \leftarrow x\hat{}i$
7: $p \leftarrow 1$
8: **Pour** $j \leftarrow 1$ **à** $n$ **faire**
9: $p \leftarrow p * b(j) * \sin((2 * j * \pi/n) * t)$
10: **fin Pour**
11: $y \leftarrow y + a(i) * p$
12: **fin Pour**
13: **fin Fonction**

**Remarque 2.** *Dans l'algorithme 2, pour optimiser le coût de calcul, le terme $x^i$ est calculé avant la boucle sur $j$, car ce terme ne dépend pas de $j$. On pourrait encore améliorer cela, en ajoutant "$t \leftarrow 1$" entre les lignes 4 et 5, puis remplaçant la ligne 7 par "$t \leftarrow x * t$". Notons que dans ce cours, sans mention contraire, il n'est pas demandé d'optimiser les algorithmes.*

2. Les données minimales sont $\boldsymbol{\alpha} = (\alpha_0, ..., \alpha_n) \in \mathbb{R}^{n+1}$, $\boldsymbol{\beta} = (\beta_0, ..., \beta_n) \in \mathbb{R}^{n+1}$, $\boldsymbol{x} = (x_0, ..., x_q) \in \mathbb{R}^{q+1}$, $p \in \mathbb{N}$ et $z \in \mathbb{R}$. Attention, il ne faut pas oublier de décaler les indices de 1 dans les vecteurs ou dans les boucles, de façon à ce que les indices des tableaux commencent bien à 1. Dans l'algorithme ci-dessous on a fait le choix de décaler les indices des vecteurs de 1. Ainsi on définit les tableaux 1D **alpha**, **beta** $\in \mathbb{R}^{n+1}$ et $\boldsymbol{X} \in \mathbb{R}^{q+1}$ tels que alpha$(k+1) = \alpha_k$, beta$(k+1) = \beta_k$ $\forall k \in [\![0, n]\!]$, et X$(j+1) = x_j$ $\forall j \in [\![0, q]\!]$.

**Algorithm 3** Fonction CalculI : Etape 1

**Données :** **alpha** $\in \mathbb{R}^{n+1}$
**beta** $\in \mathbb{R}^{n+1}$
$\boldsymbol{X} \in \mathbb{R}^{q+1}$
$p$ : nombre entier, $p \geqslant 0$
$z$ : réel

**Résultat :** $I$ : réel

1: **Fonction** $I \leftarrow$ CalculI(**alpha**, **beta**, $\boldsymbol{X}, p, z$)
2: $n \leftarrow$ length(**alpha**) $- 1$
3: $q \leftarrow$ length($\boldsymbol{X}$) $- 1$
4: $I \leftarrow 1$
5: **Pour** $k \leftarrow 0$ **à** $n$ **faire**
6: $s \leftarrow \sum_{i=0}^{p} \cos(\frac{2\pi}{k+i+1} z)$
7: $t \leftarrow \sum_{i=0}^{q} \prod_{\substack{j=0 \\ j \neq i}}^{q} \frac{z - x_j}{x_i - x_j}$
8: $I \leftarrow I *$ (alpha$(k+1) * s +$ beta$(k+1) * t$)
9: **fin Pour**
10: **fin Fonction**

$\Longrightarrow$

**Algorithm 4** Fonction CalculI : Etape 2

**Données :** **alpha** $\in \mathbb{R}^{n+1}$
**beta** $\in \mathbb{R}^{n+1}$
$\boldsymbol{X} \in \mathbb{R}^{q+1}$
$p$ : nombre entier, $p \geqslant 0$
$z$ : réel

**Résultat :** $I$ : réel

1: **Fonction** $I \leftarrow$ CalculI(**alpha**, **beta**, $\boldsymbol{X}, p, z$)
2: $n \leftarrow$ length(**alpha**) $- 1$
3: $q \leftarrow$ length($\boldsymbol{X}$) $- 1$
4: $I \leftarrow 1$
5: **Pour** $k \leftarrow 0$ **à** $n$ **faire**
6: $s \leftarrow 0$
7: **Pour** $i \leftarrow 0$ **à** $p$ **faire**
8: $s \leftarrow s + \cos(2 * \pi * z/(k + i + 1))$
9: **fin Pour**
10: $t \leftarrow 0$
11: **Pour** $i \leftarrow 0$ **à** $q$ **faire**
12: $r \leftarrow \prod_{\substack{j=0 \\ j \neq i}}^{q} \frac{x - x_j}{x_i - x_j}$
13: $t \leftarrow t + r$
14: **fin Pour**
15: $I \leftarrow I *$ (alpha$(k+1) * s +$ beta$(k+1) * t$)
16: **fin Pour**
17: **fin Fonction**

**Algorithm 5** Fonction CalculI : Etape 3

**Données :** **alpha** $\in \mathbb{R}^{n+1}$, tel que alpha$(k+1) = \alpha_k$, $\forall k \in [\![0, n]\!]$
**beta** $\in \mathbb{R}^{n+1}$, tel que beta$(k+1) = \beta_k$ $\forall k \in [\![0, n]\!]$
$\boldsymbol{X} \in \mathbb{R}^{q+1}$, tel que X$(j+1) = x_j$ $\forall j \in [\![0, q]\!]$
$p$ : nombre entier, $p \geqslant 0$
$z$ : réel

**Résultat :** $I$ : réel

1: **Fonction** $I \leftarrow$ CalculI(**alpha**, **beta**, $\boldsymbol{X}, p, z$)
2: $n \leftarrow$ length(**alpha**) $- 1$
3: $q \leftarrow$ length($\boldsymbol{X}$) $- 1$
4: $I \leftarrow 1$
5: **Pour** $k \leftarrow 0$ **à** $n$ **faire**
6: $s \leftarrow 0$
7: **Pour** $i \leftarrow 0$ **à** $p$ **faire**
8: $s \leftarrow s + \cos(2 * \pi * z/(k + i + 1))$
9: **fin Pour**
10: $t \leftarrow 0$
11: **Pour** $i \leftarrow 0$ **à** $q$ **faire**
12: $r \leftarrow 1$
13: **Pour** $j \leftarrow 0$ **à** $q$ $(j \neq i)$ **faire**
14: $r \leftarrow r * (z - X(j+1))/(X(i+1) - X(j+1))$
15: **fin Pour**
16: $t \leftarrow t + r$
17: **fin Pour**
18: $I \leftarrow I *$ (alpha$(k+1) * s +$ beta$(k+1) * t$)
19: **fin Pour**
20: **fin Fonction**

$\Longrightarrow$

**Remarque 3.** *La ligne 13 en Matlab s'écrit* `for j=[0:i-1, i+1:q]`. *Les lignes 13 à 15 de l'algorithme ci-desssus peuvent aussi être décomposées en une somme de* 0 *à* $i - 1$ *puis une somme de* $i + 1$ *à* $q$.

## Exercice 2

Ecrire une fonction DisReg générant une discrétisation régulière d'un intervalle $[a, b]$ de $\mathbb{R}$ $(a < b)$ en $n + 1$ points.

Correction

La discrétisation régulière de $[a, b]$ en $n+1$ points est définie par le vecteur $\boldsymbol{x}$ dont chaque composante $x_i$ représente un point de la discrétisation :

$$x_i = a + ih, \quad i \in [\![0, n]\!], \qquad \text{avec } h = \frac{b-a}{n},$$

et est représentée sur la figure 1.

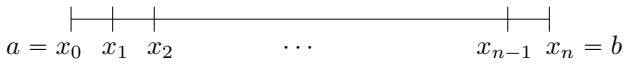



FIGURE 1 – Discrétisation régulière de $[a, b]$ en $n + 1$ points (notation mathématique)

Ici, comme dans l'exercice 1, il ne faut pas oublier de décaler les indices de 1 : on définit le tableau $\boldsymbol{X} \in \mathbb{R}^{n+1}$ tel que $X(i) = x_{i-1}$ $\forall i \in [\![1, n + 1]\!]$, voir la Figure 2

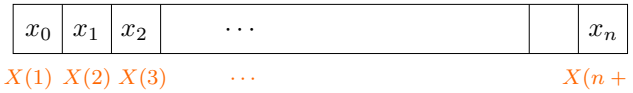



FIGURE 2 – Correspondance entre la notation mathématique $x_i$ et la notation algorithmique $X(i + 1)$

Notons que la fonction DisReg existe en MATLAB/Octave, sous le nom de "`linspace`" : $\boldsymbol{x} \leftarrow$ `linspace`$(a, b, n + 1)$.

**Algorithm 6** Fonction DisReg : calcule une discrétisation régulière d'un intervalle $[a, b]$ de $\mathbb{R}$

**Données :** $a$ : nombre réel
$b$ : nombre réel avec $b > a$
$n$ : entier , $n \geqslant 1$ (le nombre d'intervalles)

**Résultat :** $\boldsymbol{X} \in \mathbb{R}^{n+1}$ tel que $X(i) = x_{i-1} = a + (i - 1)h$, $i \in [\![1, n + 1]\!]$, avec $h = \frac{b-a}{n}$.

1: **Fonction** $\boldsymbol{X} \leftarrow$ DisReg$(a, b, n)$
2: $h \leftarrow (b - a)/n$
3: **Pour** $i \leftarrow 1$ **à** $n + 1$ **faire**
4: $X(i) \leftarrow a + (i - 1) * h$
5: **fin Pour**
6: **fin Fonction**

## Exercice 3

On considère un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$, avec $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$ inversible, et $\boldsymbol{b} \in \mathbb{R}^n$ donnés. L'unique solution de ce système est $\boldsymbol{x} = \mathbb{A}^{-1}\boldsymbol{b}$. Notons $\mathcal{L}_i$ la $i$-ème ligne de ce système. On rappelle que si l'on remplace $\mathcal{L}_i$ par une combinaison linéaire de $\mathcal{L}_i$ et d'une autre ligne $\mathcal{L}_k$ :

$$\mathcal{L}_i \leftarrow \mathcal{L}_i + \alpha\mathcal{L}_k, \quad \text{avec } \alpha \in \mathbb{R},$$

alors on obtient un système linéaire équivalent $\tilde{\mathbb{A}}\boldsymbol{x} = \tilde{\boldsymbol{b}}$, dont l'unique solution est encore $\boldsymbol{x} = \mathbb{A}^{-1}\boldsymbol{b}$.
Écrire une fonction CombLigneSys qui remplace la $i$-ème ligne d'une matrice de $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$ et d'un vecteur de $\boldsymbol{b} \in \mathbb{R}^n$ par une combinaison linéaire de la $i$-ème ligne et de la $k$-ième ligne.

Correction

**Algorithm 7** Fonction CombLignesSys :
combinaison linéaire $\mathcal{L}_i \leftarrow \mathcal{L}_i + \alpha\mathcal{L}_k$, appliquée à une matrice et un vecteur

**Données :** $\mathbb{A}$ : matrice de $\mathcal{M}_n(\mathbb{R})$,
$\boldsymbol{b}$ : vecteur de $\mathbb{R}^n$,
$i, k$ : entiers, $1 \leqslant i, k \leqslant n$
$\alpha$ : réel

**Résultat :** $\mathbb{A}$ et $\boldsymbol{b}$ modifiés , obtenus en remplaçant leur $i$-ème ligne $\mathcal{L}_i$, par $\mathcal{L}_i + \alpha\mathcal{L}_k$.

1: **Fonction** $[\mathbb{A}, \boldsymbol{b}] \leftarrow$ CombLignesSys$(\mathbb{A}, \boldsymbol{b}, i, k, \alpha)$
2: $n \leftarrow$ `length`$(\boldsymbol{b})$
3: **Pour** $j \leftarrow 1$ **à** $n$ **faire**
4: $\mathbb{A}(i, j) \leftarrow \mathbb{A}(i, j) + \alpha * \mathbb{A}(k, j)$
5: **fin Pour**
6: $\boldsymbol{b}(i) \leftarrow \boldsymbol{b}(i) + \alpha * \boldsymbol{b}(k)$
7: **fin Fonction**

Notons qu'ici on retourne directement $\mathbb{A}$ et $\boldsymbol{b}$ modifiés, ce qui évite le stockage de $\tilde{\mathbb{A}}$ et $\tilde{\boldsymbol{b}}$ et surtout la recopie intégrale de la matrice et du vecteur.

### Exercice 4

On considère un système linéaire $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$, avec $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$ inversible, et $\boldsymbol{b} \in \mathbb{R}^n$ donnés. Si l'on permute la $k$-ème ligne $\mathcal{L}_k$ avec la $\ell$-ème ligne $\mathcal{L}_\ell$, on retrouve le même système linéaire.
Écrire une fonction PermLignesSys qui permute deux lignes $k$ et $\ell$ d'une matrice $\mathbb{A}$ de $\mathcal{M}_n(\mathbb{R})$ et d'un vecteur de $\mathbb{R}^n$, avec $k, \ell \in [\![1, n]\!],\ k \neq \ell$.

Correction

**Algorithm 8** Fonction PermLignesSys : permute 2 lignes $k$ et $\ell$ d'une matrice et d'un vecteur

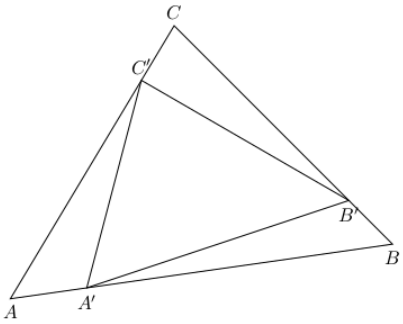
**Données :** $\mathbb{A}$ : matrice de $\mathcal{M}_n(\mathbb{R})$,
$\boldsymbol{b}$ : vecteur de $\mathbb{R}^n$,
$k, \ell$ : entiers, $1 \leqslant k, \ell \leqslant n$

**Résultat :** $\mathbb{A}$ et $\boldsymbol{b}$ modifiés

1: **Fonction** $[\mathbb{A}, \boldsymbol{b}] \leftarrow$ PermLignesSys$(\mathbb{A}, \boldsymbol{b}, k, \ell)$
2: $n \leftarrow$ `length`$(\boldsymbol{b})$
3: **Pour** $j \leftarrow 1$ **à** $n$ **faire**
4: $t \leftarrow \mathbb{A}(k, j),\ \mathbb{A}(k, j) \leftarrow \mathbb{A}(\ell, j),\ \mathbb{A}(\ell, j) \leftarrow t$
5: **fin Pour**
6: $t \leftarrow \boldsymbol{b}(k),\ \boldsymbol{b}(k) \leftarrow \boldsymbol{b}(\ell),\ \boldsymbol{b}(\ell) \leftarrow t$
7: **fin Fonction**

Notons qu'ici, comme pour la fonction CombLignesSys, on retourne directement $\mathbb{A}$ et $\boldsymbol{b}$ modifiés pour éviter la recopie intégrale de la matrice et du vecteur.

### Exercice 5




Soit $T$ un triangle de sommets $A, B$ et $C$. À partir de ce triangle on peut construire un nouveau triangle de sommets $A', B'$ et $C'$ vérifiant

$$\overrightarrow{AA'} = x\overrightarrow{AB}$$
$$\overrightarrow{BB'} = x\overrightarrow{BC}$$
$$\overrightarrow{CC'} = x\overrightarrow{CA}$$

avec $x$ un réel strictement compris entre 0 et 1.

Écrire une fonction triangles permettant à partir des trois sommets $A, B, C$ d'un triangle initial quelconque non réduit à une droite ou un point, de représenter ce triangle ainsi que les $n$ triangles obtenus par le processus de construction décrit ci-dessus avec un $x$ donné dans $]0, 1[$. On dispose pour celà de la fonction plot$([x_A, x_B], [y_A, y_B])$ permettant de tracer le segment $[A, B]$ du plan avec $A = (x_A, y_A)$ et $B = (x_B, y_B)$. Correction

On commence par écrire une fonction PlotTriangles qui trace un triangle, à partir de ses trois sommets, en utilisant la fonction plot. Ensuite on écrit la fonction CalculPts qui calcule le nouveau point $A', B', C'$ à partir de $A, B, C$. Enfin, on écrit la fonction triangles qui utilise de façon itérative les fonctions PlotTriangles et CalculPts.

**Algorithm 9** Fonction PlotTriangles : représentation graphique d'un triangle de sommets $A, B$ et $C$

**Données :** $A$ : vecteur de $\mathbb{R}^2$,
$B$ : vecteur de $\mathbb{R}^2$,
$C$ : vecteur de $\mathbb{R}^2$

1: **Fonction** PlotTriangles$(A, B, C)$
2: plot$([A(1), B(1)], [A(2), B(2)])$
3: plot$([B(1), C(1)], [B(2), C(2)])$
4: plot$([C(1), A(1)], [C(2), A(2)])$
5: **fin Fonction**

**Algorithm 10** Fonction CalculPts : calcule les points $A', B', C'$ à partir des points $A, B, C$ par le processus de l'exercice 5

**Données :** $A, B, C$ : vecteur de $\mathbb{R}^2$,
$x$ : réel dans $]0, 1[$

**Résultat :** $Ap, Bp, Cp$ : vecteurs de $\mathbb{R}^2$ modifiés

1: **Fonction** $[Ap, Bp, Cp] \leftarrow$ CalculPts$(A, B, C, x)$
2: **Pour** $i \leftarrow 1$ **à** $2$ **faire**
3: $Ap(i) \leftarrow A(i) + x * (B(i) - A(i))$
4: $Bp(i) \leftarrow B(i) + x * (C(i) - B(i))$
5: $Cp(i) \leftarrow C(i) + x * (A(i) - C(i))$
6: **fin Pour**
7: **fin Fonction**

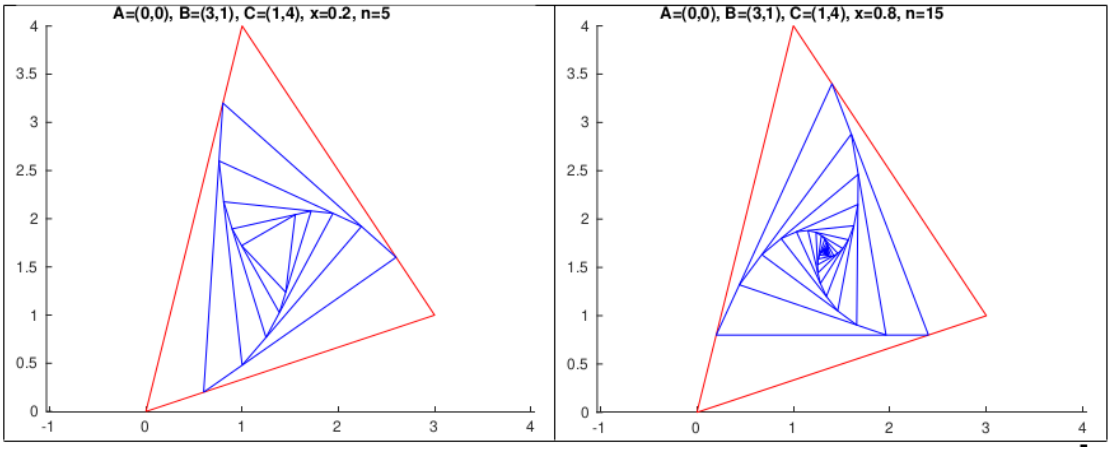
$\Rightarrow$

**Algorithm 11** Fonction TRIANGLES : représente graphiquement un triangle $A, B, C$ ainsi que les $n$ triangles obtenus à partir de ce triangle, par le processus de l'exercice 5

**Données :** $A$ : vecteur de $\mathbb{R}^2$,
$B$ : vecteur de $\mathbb{R}^2$,
$C$ : vecteur de $\mathbb{R}^2$
$x$ : réel dans $]0, 1[$
$n$ : entier , $n \geqslant 1$

1: **Fonction** TRIANGLES$(A, B, C, x, n)$
2: PlotTriangles$(A, B, C)$
3: **Pour** $i \leftarrow 1$ **à** $n$ **faire**
4: $[A, B, C] \leftarrow$ CalculPts$(A, B, C, x)$
5: PlotTriangles$(A, B, C)$
6: **fin Pour**
7: **fin Fonction**

Voici deux exemples d'utilisation de cette fonction :




**Remarque 4.** *Notons que dans l'algorithme 10, si en sortie on souhaite mettre $[A, B, C]$, on aura quand même besoin d'utiliser les variables temporaires $Ap, Bp, Cp$. En effet, si l'on remplace les lignes 3,4,5 de l'algorithme directement par*

$$A(i) \leftarrow A(i) + x * (B(i) - A(i)) \quad (1)$$
$$B(i) \leftarrow B(i) + x * (C(i) - B(i)) \quad (2)$$
$$C(i) \leftarrow C(i) + x * (A(i) - C(i)) \quad (3)$$

*l'algorithme ne donnera pas le résultat demandé. Ceci est dû au fait que la ligne (1) ci-dessus modifie la valeur de $A(i)$ et va donner ainsi une mauvaise valeur de $A(i)$ à la ligne (3) ci-dessus, lors du calcul de $C(i)$. Pour que cela soit correct on ajoute, à la suite des lignes 2 à 6, les lignes 7 à 11 ci-dessous :*

1: **Fonction** $[A, B, C] \leftarrow$ CalculPts$(A, B, C, x)$
2: **Pour** $i \leftarrow 1$ **à** $2$ **faire**
3: $Ap(i) \leftarrow A(i) + x * (B(i) - A(i))$
4: $Bp(i) \leftarrow B(i) + x * (C(i) - B(i))$
5: $Cp(i) \leftarrow C(i) + x * (A(i) - C(i))$
6: **fin Pour**
7: **Pour** $i \leftarrow 1$ **à** $2$ **faire**
8: $A(i) \leftarrow Ap(i)$
9: $B(i) \leftarrow Bp(i)$
10: $C(i) \leftarrow Cp(i)$
11: **fin Pour**
12: **fin Fonction**