2de TD n°5 : utilisation de la calculatrice pour déterminer un extremum

1) Balayage d'un intervalle

Nous allons écrire un algorithme qui demande les bornes a et b d'un intervalle $[a;b]$ et qui détermine et affiche les images de valeurs régulièrement espacées dans cet intervalle pour une fonction enregistrée de la calculatrice.

Pour découper l'intervalle $[a;b]$ en dix, on calcule le « pas » $p=\frac{b-a}{10}$ pour augmenter la variable. On calcule et on affiche ensuite les images des nombres :

$a,\ a+p,\ a+2p,\ \ldots,\ b$ (ce qui fait 11 nombres).

Saisir A
Saisir B
P=(B−A)/10
Pour I allant de 0 à 10
X=A+I×P
Afficher f(X)
fin de la boucle « Pour »

➢ Programmer cet algorithme « BALAYAGE » sur votre calculatrice.

Algorithme	Programme en Basic Casio	Programme en Basic TI
Saisir A Saisir B P=(B−A)/10 Pour I allant de 0 à 10 X=A+I×P Afficher f(X) fin de la boucle « Pour »	?→A ?→B (B−A)/10→P For 0→I To 10 A+I×P→X Y1◢ Next	Input A Input B (B−A)/10 STO→P For (I,0,10) A+I×P STO→X Disp Y1 End

Commentaire 1 : Il y a de multiples façons de réaliser cet algorithme. Nous en avons juste présenté une. Le choix de la boucle « Pour » paraît tout indiqué, puisque l'on sait combien de tours de boucle il faut faire : 11 tours. Ce sera réalisé en prenant I entre 0 et 10. On aurait pu réaliser une boucle « Tant que » en écrivant « Tant que A+I×P≤B », mais il aurait fallu gérer la variable I (initialisation avant d'entrer dans la boucle, puis incrémentation dans la boucle).

Commentaire 2 : Le X est celui de la touche spéciale X,θ,t et la variable Y1 se trouve dans un dossier (VAR, voir la fin de la feuille de TD n°4).

Commentaire 3 : Si vous voulez utiliser ce programme pour une autre fonction, il suffit de changer dans le module de saisie des fonctions. Le programme reste inchangé (c'est son avantage). Vous devrez aussi généralement changer les valeurs de A et B (les bornes de l'intervalle).

➢ Compléter le tableau de données pour la fonction f définie par $f(x)=2x^4-5x^3+1$ sur [0 ; 3].

Pour calculer les images de X par f, on doit enregistrer la fonction en Y1. Ici on entrera A=0 et B=3.

x	0	0,3	0,6	0,9	1,2	1,5	1,8	2,1	2,4	2,7	3
$f(x)$	1	0,881	0,179	-1,333	-3,493	-5,75	-7,165	-6,409	-1,765	8,873	28,0

Remarque : il existe, on l'a vu, une autre façon d'obtenir un tableau de données pour une fonction enregistrée de la calculatrice.

Voici, pour ceux que cela intéresse le même algorithme en Python et la sortie en bleu (à droite)

```
# fonction du TD n°5 (ex 1)
def image():
    return 2*x**4-5*x**3+1

a=int(input("Entrer le nombre a: "))
b=int(input("Entrer le nombre b: "))
c=10 # nombre de parties
pas=(b-a)/c
for i in range(c+1):
    x=a+i*pas
    print("x= {} - y= {}".format(round(x,1),round(image(),3)))
```

```
Entrer le nombre a: 0
Entrer le nombre b: 3
x= 0.0 - y= 1.0
x= 0.3 - y= 0.881
x= 0.6 - y= 0.179
x= 0.9 - y= -1.333
x= 1.2 - y= -3.493
x= 1.5 - y= -5.75
x= 1.8 - y= -7.165
x= 2.1 - y= -6.409
x= 2.4 - y= -1.765
x= 2.7 - y= 8.873
x= 3.0 - y= 28.0
```

2) Recherche de l'extremum

a) On peut modifier légèrement cet algorithme de balayage pour rechercher le minimum d'une fonction sur $[a;b]$

On utilise deux nouvelles variables m et n (m sera le minimum obtenu et n sera la valeur de x pour laquelle ce minimum est atteint) qu'on initialise à $m=f(a)$ et $n=a$, puis on balaye l'intervalle $[a\ ;\ b]$ sans afficher les valeurs mais en testant si $f(x)<m$. Si c'est le cas, on remplace les valeurs de m et n par $m=f(x)$ et $n=x$. On affiche alors ces deux valeurs.

Remarque : le découpage de l'intervalle en 10 parties ne permet pas de résultats très précis. Si on veut déterminer la valeur n (valeur de la variable x pour laquelle ce minimum est atteint) à la précision 10^{-2}, il faut que $p\leq 10^{-2}$ et donc, il faut découper l'intervalle en, au moins, $\frac{b-a}{10^{-2}}$ soit $10^2\times(b-a)$ parties.

➢ Modifier votre programme « BALAYAGE » pour réaliser cet objectif.

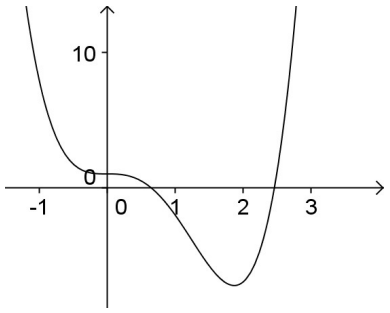
Il faut initialiser M et N avant d'entrer dans la boucle ; tester $f(X)$ dans la boucle pour changer les valeurs de M et N si nécessaire (si $f(X)<M$) ; afficher M et N après la boucle. On peut garder l'affichage des valeurs mais je vais choisir de supprimer cet affichage. Voici l'algorithme modifié et les programmes correspondants.

Algorithme	Programme en Basic Casio	Programme en Basic TI
Saisir A Saisir B P=(B−A)/10 X=A M=f(X) N=A Pour I allant de 0 à 10 X=A+I×P Si f(X)<M Alors M=f(X) N=X fin du « si » fin de la boucle « Pour » Afficher M Afficher N	?→A ?→B (B−A)/10→P A→X Y1→M A→N For 0→I To 10 A+I×P→X If Y1<M Then Y1→M A→N IfEnd Next M◢ N◢	Input A Input B (B−A)/10 ^{STO}→P A^{STO} →X Y1^{STO} →M A^{STO} →N For (I,0,10) A+I×P ^{STO}→X If Y1<M Then Y1^{STO}→M A^{STO}→N End End Disp M Disp N

Commentaire 1 :Je ne suis pas sûr de la syntaxe correcte pour Then. Peut-être que sur certaines calculatrices, il faut (ou on peut) entrer « Then Y1→M » directement (sur la même ligne).

Si on exécute ce programme modifié, la calculatrice affiche -7,165 puis 1,8 puisque ce sont les valeurs obtenues de M et de N pour le minimum.

Commentaire 2 : Si vous voulez utiliser ce programme pour déterminer un maximum au lieu du minimum, il suffit de change le test « Si f(X)<M » en « Si f(X)>M ».



➢ Déterminer le minimum et son antécédent à 10^{-2} près. D'après notre remarque, il faut découper l'intervalle en $10^2\times(3-0)=300$ parties, donc écrire $p=\frac{b-a}{300}$.

Si on change la division en 300 au lieu de 10, il y a deux endroits à modifier. Vous pouvez mettre le nombre de parties dans une variable C, initialisée à 10 ou à 300 ici. Au lieu d'avoir à changer les deux endroits à chaque fois que vous changer ce nombre, vous mettez C à la place de 10.

Algorithme

Saisir A
Saisir B
C=300
P=(B−A)/**C**
X=A
M=f(X)
N=A
Pour I allant de 0 à **C**
X=A+I×P
Si f(X)<M
Alors
M=f(X)
N=X
fin du « si »
fin de la boucle « Pour »
Afficher M
Afficher N

cet algorithme approximativement programmé en Python

```
# fonction du TD n°5 (ex 2)
def image():
    return 2*x**4-5*x**3+1

a=int(input("Entrer le nombre a: "))
b=int(input("Entrer le nombre b: "))
c=int(input("Entrer le nombre de subdivisions : ")) # nombre de parties
pas=(b-a)/c
n=a
x=a
m=image()
for i in range(c+1):
    x=a+i*pas
    if image()<m:
        n=x
        m=image()
print("le pas est égal à {}".format(round(pas,6)))
print("le minimum est environ {}, atteint pour x environ {}".format(round(m,6),round(n,3)))
```

```
Entrer le nombre a: 0
Entrer le nombre b: 3
Entrer le nombre de subdivisions : 300
le pas est égal à 0.01
le minimum est environ -7.239396, atteint pour x environ 1.87
```

Résultat : $n=1{,}87\pm10^{-2}$; $m=f(n)$=-7,239396

b) Pour obtenir une meilleure précision sans faire trop de calculs, on doit améliorer l'efficacité de cet algorithme de balayage. Il faut découper l'intervalle [0 ; 3] en $10^6\times(3-0)=3000000$ parties pour obtenir la valeur de n à 10^{-6} près. C'est beaucoup. On va écrire une boucle « While » pour limiter ce nombre.

On découpe l'intervalle en 10 parties et on lance le balayage. Cela permet d'encadrer la valeur de n : $n-p<n<n+p$. On recommence alors en modifiant les bornes de l'intervalle : $a=n-p$ et $b=n+p$, et en modifiant p en conséquence (p sera toujours $\frac{b-a}{10}$ mais avec les nouvelles valeurs de a et b). On arrête le programme par le test de la boucle *while* : tant que $p> 10^{-6}$ si on veut une précision de n à 10^{-6} près.

➢ Écrire un programme « MINIMUM » pour réaliser cet objectif.

Algorithme	Programme en Basic Casio	Programme en Basic TI
Saisir A Saisir B **Tant que (B−A)/10>0,000001** P=(B−A)/10 X=A M=*f*(X) N=A Pour I allant de 0 à 10 X=A+I×P Si *f*(X)<M Alors M=*f*(X) N=X fin du « si » fin de la boucle « Pour » **A=N-P** **B=N+P** **fin de la boucle « Tant que »** Afficher M Afficher N	?→A ?→B **While (B−A)/10>0,000001** (B−A)/10→P A→X Y1→M A→N For 0→I To 10 A+I×P→X If Y1<M Then Y1→M A→N IfEnd Next **N-P→A** **N+P→B** **WEnd** M◢ N◢	Input A Input B **While (B−A)/10>0,000001** (B−A)/10 ^{STO}→P A^{STO} →X Y1^{STO} →M A^{STO} →N For (I,0,10) A+I×P ^{STO}→X If Y1<M Then Y1^{STO}→M A^{STO}→N End End **N-P ^{STO} →A** **N+P ^{STO} →B** **End** Disp M Disp N

Commentaire 1 : Il y a ici deux boucles imbriquées l'une dans l'autre. La boucle « While » est la plus extérieure. Il faut veiller au test de sortie (**(B−A)/10>0,000001**) pour que l'on puisse entre dans la boucle au départ. B et A doivent être modifiés à la fin de la boucle avec les valeurs retenues dans le texte.

➢ Déterminer le minimum et son antécédent à 10^{-6} près.

Résultat : $n=1{,}874999\pm10^{-6}$; $m=f(n)=-7{,}239746$

c) Combien d'appels de la fonction sont nécessaires ici ?

Pour le savoir, introduisez une nouvelle variable I, initialisée à 0 et augmentée de 1 à chaque appel de fonction. Vous devriez obtenir un I beaucoup plus petit que 3000000. Comparer les performances des deux algorithmes sur le nombre d'appel de la fonction et le temps d'exécution :

```
# fonction du TD n°5 (ex 3)
def image():
    return 2*x**4-5*x**3+1

a=int(input("Entrer le nombre a: "))
b=int(input("Entrer le nombre b: "))
c=10 # nombre de parties
f=0  # nombre d'appels de la fonction
while (b-a)/c>0.000001:
    pas=(b-a)/c
    n=a
    x=a
    m=image()
    f=f+1
    for i in range(c+1):
        x=a+i*pas
        f=f+1
        g=image()
        if g<m:
            n=x
            m=g
    a=n-pas
    b=n+pas
print("le minimum est environ {}, atteint pour x environ {} après {} appels".format(round(m,6),round(n,6),f))
```

```
Entrer le nombre a: 0
Entrer le nombre b: 3
le minimum est environ -7.239746, atteint pour x environ 1.874999 après 96 appels
```

	Nombre d'appels de la fonction	Durée	Bilan
Méthode a : un seul grand balayage	3000000	28*h*	Trop long !
Méthode b : balayage sélectif	96	<1*s*	Très efficace !

Commentaire 1 :La durée est proportionnelle au nombre de tours de boucle. Si il faut 10*s* pour la question précédente (pour 300 subdivisions) pour 3 000 000subdivisions (10 000 fois plus) il faudra 100 000*s* soit près de 28*h*... C'est beaucoup trop long. Ne lancez pas votre calculatrice pour cela sur une durée aussi longue. Par contre l'ordinateur va beaucoup plus vite. Programmé en Python, cet algorithme met environ 10s pour 3 000 000 de subdivisions.

```
Entrer le nombre a: 0
Entrer le nombre b: 3
Entrer le nombre de subdivisions : 3000000
le pas est égal à 1e-06
le minimum est environ -7.239746, atteint pour x environ 1.875
```

Commentaire 2 : Pour introduire un compteur, nous devons l'initialiser à 0 avant d'entrer dans la boucle, puis il faut l'incrémenter (ajouter 1) à chaque appel de fonction. Pour minimiser ce nombre, nous avons utilisé une mémoire supplémentaire G pour mettre *f*(X) car on en a besoin deux fois (une fois dans le test du « If » et l'autre fois dans l'affectation de *f*(X) dans M si le test est positif). Sans cette précaution, vous obtiendrez une plus grande valeur à la fin.