Algorithme de balayage pour la recherche d'un extremum

Enoncé

Soit f la fonction définie sur $\mathbb{R}$ par : $f(x) = -2x^2 + 6x + 8$. On souhaite rechercher le maximum de cette fonction. On va utiliser deux méthodes différentes.

1. Première méthode : retranscrire, en Python, la fonction **Maximum** ci-contre. On suppose que a < b et **f** est la fonction Python qui renvoie l'image d'un réel par f. A l'aide la fonction **Maximum**, utilisée avec les paramètres pertinents, déterminer le maximum de la fonction f.
2. Deuxième méthode : à l'aide de la calculatrice, déterminez les valeurs x_1 et x_2 pour lesquels la fonction f s'annule et en déduire la valeur du maximum, connaissant les propriétés de la fonction f.

```
Fonction Maximum(a,b,p)
      xmax ← a
      ymax ← f(a)
      Tant que x < b
            Si f(x) > f(xmax) alors
                  xmax, ymax ← x, f(x)
            Fin Si
      x ← x + p
      Fin Tant que
      Renvoyer xmax, ymax
```

1. Méthode avec Python

On crée un script **BALAYAGE** dans lequel on commence par définir la fonction Python **f** qui renvoie l'image de x par f à l'aide de l'instruction :
`return -2*x**2 + 6*x + 8`

On définit ensuite la fonction **Maximum** de paramètres **a**,**b** et **p**.

Il existe plusieurs façons de réaliser un balayage. Ici, on a fixé le pas, c'est-à-dire la valeur avec laquelle on augmente **x** à chaque itération. On aurait pu opter pour passer en paramètre le nombre de valeurs de **x** testées.

On utilise donc une boucle **while**. Tant que **x**, notre variable intermédiaire, est inférieure à **b**, on teste si son image par la fonction **f** est plus grande que l'image maximale enregistrée. Si oui, on affecte les nouvelles valeurs. Dans tous les cas, on poursuit notre balayage jusqu'à dépasser **b**.

Il faut penser à initialiser les variables **xmax** et **ymax** qui seront renvoyées par la fonction.

En mode console, on commence par évaluer quelques images de la fonction **f** pour avoir une idée où rechercher notre maximum.

On a $f(-3) = -28$, $f(1) = 12$ et $f(5) = -12$. On teste notre fonction à l'aide de l'instruction **Maximum(-3,5,0.1)**.

On obtient un maximum pour $x = 1{,}5$. On évalue notamment $f(1{,}5)$ pour comprendre qu'il s'agit bien d'un affichage, en quelque sorte, arrondi par Python.

Une question qui peut se poser est comment faire maintenant pour rechercher le maximum d'une autre fonction ? Par exemple, le maximum de la fonction h définie sur $\mathbb{R}$ par : $h(x) = -2(x+1)(x-6)$

On peut bien sûr redéfinir dans notre script la fonction f. Mais il est possible de rendre notre fonction **Maximum** plus « universelle » en lui ajoutant en paramètre le nom de la fonction dont on souhaite rechercher le maximum par balayage.

ÉDITEUR : BALAYAGE
LIGNE DU SCRIPT 0003

```
# Calculs Mathématiques
from math import *

def f(x):
  return -2*x**2+6*x+8
```

ÉDITEUR : BALAYAGE
LIGNE DU SCRIPT 0016

```
def Maximum(a,b,p):
  xmax = a
  ymax = f(a)
  x = a
  while x < b :
    if f(x)>f(xmax):
      xmax,ymax = x,f(x)
    x += p
  return [xmax,ymax]
```

Fns… | a A # | Outils | Exéc | Script

PYTHON SHELL

```
>>> f(-3)
-28
>>> f(1)
12
>>> f(5)
-12
>>> Maximum(-3,5,0.1)
[1.500000000000002, 12.5]
>>> f(1.5)
12.5
>>>
```

Fns… | a A # | Outils | Éditer | Script

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

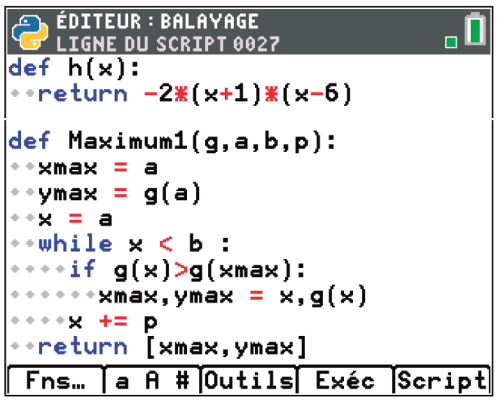
© Texas Instruments 2022 / Photocopie autorisée

Algorithme de balayage pour la recherche d'un extremum

Créons une fonction **h** et une fonction **Maximum1** selon le script ci-contre.

Afin de faciliter la compréhension, on a appelé **g** la fonction passée en paramètre. **Maximum1** est proche du code de **Maximum** mais maintenant nous pouvons utiliser différentes fonctions dans notre script et ne plus dépendre de la seule fonction **f**.

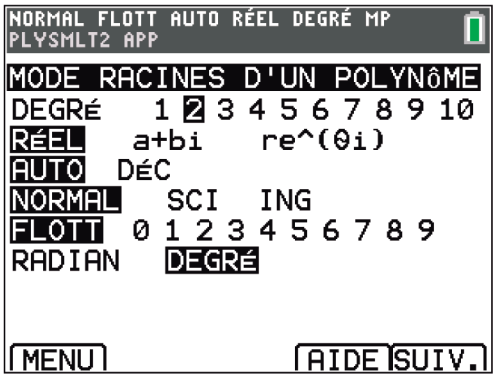
```
>>> Maximum1(f,-3,5,0.1)
[1.500000000000002, 12.5]
>>> Maximum1(h,-3,5,0.1)
[2.500000000000003, 24.5]
>>>
```

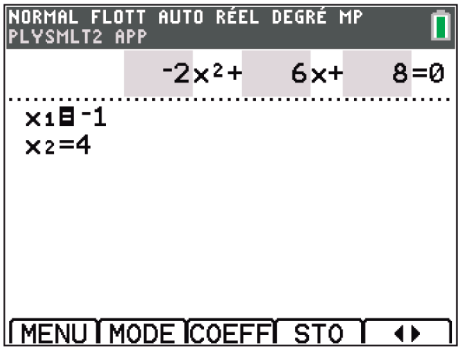


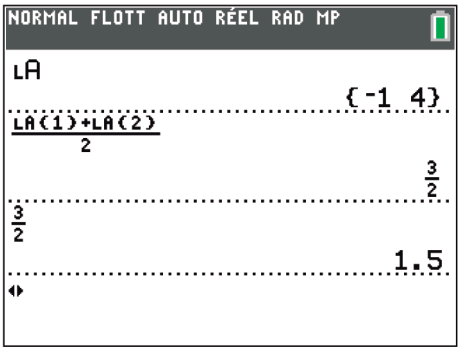
2. Méthode sans Python

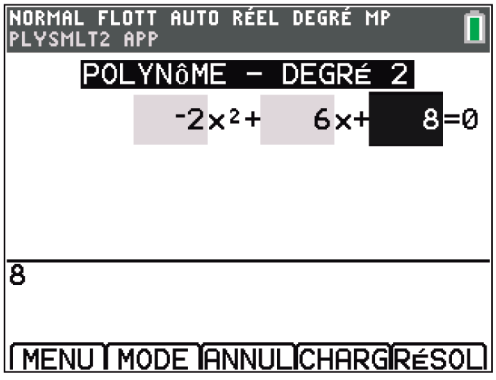
La calculatrice dispose de plusieurs outils pour ce genre d'exploration de fonctions. En particulier, elle dispose d'une application **PlySmlt2** capable de donner les racines d'un polynôme de degré 2. Nous allons nous en servir dans le cas de notre travail. D'autant que les solutions peuvent être exportées dans les listes pour être ensuite exploitées dans des calculs.

On configure l'application pour trouver les racines d'un polynôme de degré 2, que l'on saisit ensuite. L'onglet RÉSOL permet d'obtenir les racines, que l'on sauvegarde dans une liste **LA** à l'aide de l'onglet STO et rappelée dans l'écran de calcul à l'aide du menu **listes** (2nde + stats) puis **A.**



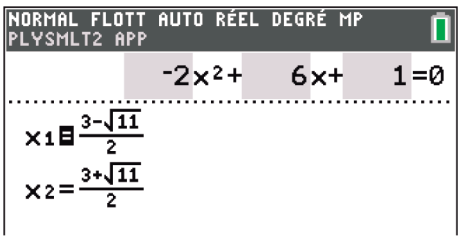


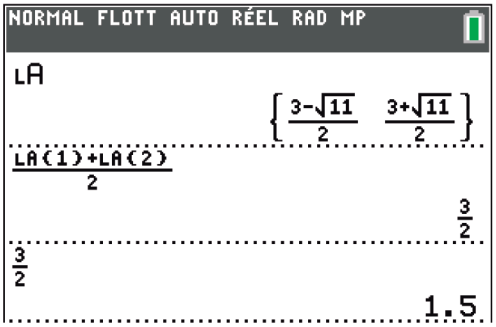




Dans le cas du polynôme f ou h, les racines sont suffisamment simples pour se passer de cette manipulation mais cela devient intéressant par exemple avec la fonction définie sur $\mathbb{R}$ par : $g(x) = -2x^2 + 6x + 1$.

En effet, de par les propriétés de symétrie, le maximum (ou minimum selon la nature du trinôme) est atteint pour la valeur moyenne des racines. Celles-ci peuvent parfois être longues à saisir.





Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

© Texas Instruments 2022 / Photocopie autorisée