Programmes TI-Voyage 200[1]

Résolution d'une équation de second degré

```
second2()
Prgm
ClrIO
Local delta,a,b,c,x1,x2,x,y,z
Lbl Label
Dialog
Title "Equation second degré"
Text "ax^2 + bx + c = 0"
Request "valeur de a, a > 0",x
Request "valeur de b",y
Request "valeur de c",z
EndDlog
expr(x) → a
expr(y) → b
expr(z) → c
If ok=0 : stop
If a ≤ 0 Then
Goto label
b*b-4*a*c → delta
If delta > 0 Then
(-b-√(delta))/(2*a)→ x1
(-b+√(delta))/(2*a)→ x2
Disp "solution réelle x1 :",x1
Disp "solution réelle x2 :",x2
EndIf
If delta=0 Then
-b/(2*a)→ x1
Disp "solution double x1 :",x1
EndIf
If delta < 0 Then
(-b-i*√(delta))/(2*a)→ x1
(-b+i*√(delta))/(2*a)→ x2
Disp "solution imaginaire x1 :",x1
Disp "solution imaginaire x2 :",x2
EndIf
Pause : Disphome
EndPrgm
```

Résolution d'une équation de second degré (plus simple)

```
second2()
Prgm
ClrIO
Local delta,a,b,c,x1,x2
Disp "ax^2 + bx + c = 0"
Input "a= ?",a
Input "b= ?",b
Input "c= ?",b
b*b-4*a*c → delta
If delta > 0 Then
(-b-√(delta))/(2*a)→ x1
(-b+√(delta))/(2*a)→ x2
Disp "solution réelle x1 :",x1
Disp "solution réelle x2 :",x2
EndIf
If delta=0 Then
-b/(2*a)→ x1
Disp "solution double x1 :",x1
EndIf
If delta < 0 Then
(-b-i*√(delta))/(2*a)→ x1
(-b+i*√(delta))/(2*a)→ x2
Disp "solution imaginaire x1 :",x1
Disp "solution imaginaire x2 :",x2
EndIf
Pause : Disphome
EndPrgm
```

[1] Réalisé par Gwendal Haudebourg avec LaTeX. Mise à jour le 31/07/2006.

Le nombre n est-il premier ?

```
prem()
Prgm
Local p,n,d
ClrIO
1 → p
2 → d
Lbl demande
Input "Rentrez un nombre entier > 1 ",n
If n ≤ 1 Then
Goto demande
EndIf
While d*d ≤ n
If mod(n,d)=0 Then
0 → p
EndIf
d+1 → 1
EndWhile
If p=1 Then
Disp "Le nombre est premier"
Else
Disp "Le nombre n'est pas premier"
EndIf
Pause : DispHome
EndPrgm
```

remarque : on peut modifier la boucle par : While d*d ≤ n and p ≠ 0, cela va beaucoup plus vite

Afficher n nombres premiers

```
premn()
Func
Local d,p,k,L1
{2} → L1
3 → k
While k ≤ n
1 → p
2 → d
While d*d ≤ k
If mod(k,d)=0 Then
0 → p
EndIf
d+1 → d
EndWhile
If p=1 Then
augment({k},L1) → L1
EndIf
k+1 → k
EndWhile
Return L1
EndFun
```

Bezout : nu+pv=1

```
Bezout()
Prgm
Local q,r,a,b,u,v,t
ClrI0
Input “n= ?”,n
Input “p= ?”,p
If n>p Then
n→ a : p→ b
Else
n→ b : p→ a
EndIf
1 → r
While r
neq 0
mod(a,b) → r
b → a : r → b
EndWhile
Disp “PGCD=”,a
0 → i
If a=1 Then
Prompt t
For u,-t-t
For v,-t-t
If n*u+p*v=1 Then
i+1 → i
u → list1[i]
v → list2[i]
EndIf
EndFor
EndFor
EndIf
Pause : DispHome
EndPrgm
```

Pour afficher les solutions : liste1 (puis entrée) et liste2(puis entrée). Cet algorithme permet de trouver des couples (u,v) solutions de nu+pv=1, lorsque pgcd(n,p)=1. Il est extrêmement lent. La variable t représente l'intervalle de solutions de u et v.
Exemple : Bezout() pour n=8,p=11 puis t=7 :
list1 renvoie $\{-4, 7\}$, list2 renvoie $\{3, -5\}$, donc les couples (-4,3) et (7,-5) sont solutions.

PGCD de deux nombres

```
pgcd()
Prgm
Local q,r,a,b
ClrI0
Input “n= ?”,n
Input “p= ?”,p
If n>p Then
n→ a : p→ b
Else
n→ b : p→ a
EndIf
1 → r
While r ≠ 0
mod(a,b) → r
b → a : r → b
EndWhile
Disp ”PGCD=”,a
Pause : DispHome
EndPrgm
```

Méthode d'Euler 1

Programme pour la TI en langue anglaise : résolution graphique d'une équation différentielle par la méthode d'Euler (pris dans Initiation Voyage 200, P.7). Les "@**...**" sont des commentaires. Pour le mode français : remplacer Newlist par NouvList, et Zoomdata par ZoomDonn. Le programme qui suit est la méthode d'euler explicite, pour la fonction $f(x) = x + 1$, c'est à dire pour l'équation différentielle $y' = y + 1$, avec un pas de 0.1, et pour 40 points.

euler()
Prgm
Local f,x0,y0,p,n,lx,ly,i
"x+1"→ f :"0"→ x0 :"0"→ y0 :"0.1"→ p
"40"→ n
@ **la ligne précédente sert à mettre des valeurs par défaut pour le programme**
Dialog
Title "Entrée des données"
Request "f(x)",f
Request "x0 ",x0
Request "y0 ",y0
Request "Pas ",p
Request "Pts ",n
Endlog
expr(f)→ f : expr(x0)→ x0 : expr(y0)→ y0
expr(p)→ p : expr(n)→ n
Newlist(n)→ lx : Newlist(n)→ ly
For i,1,n
x0 → lx[i] :y0 → ly[i]
y0+p*(f| x=x0)→ y0 :x0+p→ x0
EndFor
lx → liste1 :ly → liste2
Newplot 1,2,liste1,liste2,,,,3
Zoomdata @ **pour zoomer sur les données. Pas nécessaire, et même parfois troublant**
EndPrgm

On trouve dans les manuels de TI une comparaison entre Euler et Runge-Kutta : P.11-19 pour le manuel de la TI 89, p.686 pour le manuel TI 89-Voyage 200.

Méthode d'Euler 2

L'exemple qui suit est la méthode d'euler explicite, pour la fonction $f(t) = a(t)y + b(t)$. On peut tester par exemple $a(t) = -t$ et $b(t) = 1$, ie équation différentielle $y' = -y + 1$, avec un pas de 0.1, et pour 40 points.

euler2()
Prgm
Local a,b,x0,y0,p,n,lx,ly,i
Dialog
Title "Entrée des données"
Request "a(x)",a
Request "b(x)",b
Request "x0 ",x0
Request "y0 ",y0
Request "Pas ",p
Request "Pts ",n
EndDlog
expr(a)→ a : expr(b)→ b : expr(x0)→ x0 :
expr(y0)→ y0
expr(p)→ p : expr(n)→ n
Newlist(n)→ lx : Newlist(n)→ ly
For i,1,n
x0 → lx[i] :y0 → ly[i]
(y0+p*(b| x=x0))/(1-p*(a|x=x0)→ y0 x0+p→ x0
EndFor
lx → liste3 :ly → liste4
Newplot 2,2,liste3,liste4,,,,3
@ **qques petits chgts pour ne pas écraser le travail réalisé avec Euler 1**
Zoomdata
EndPrgm

Méthode de la Dichotomie

```
dicho()
Prgm
ClrIO
Local a,b,n,m,p,i
Input "borne inferieure de [a,b]",a
Input "borne superieure de [a,b]",b
Input "precision voulue",p
a → m
b → n
o → i
While abs(m-n)>p
If y1(m)<0 Then
If y1((m+n)/2)>0 Then
(m+n)/2 → n
Else
(m+n)/2 → m
EndIf
Else If y1((m+n)/2)<0 Then
(m+n)/2 → n
Else
(m+n)/2 → m
EndIf
EndIf
i+1 → i
EndWhile
Disp "la solution est dans l'intervalle",
[[round(m),round(n)]]
Disp "nombre iterations :",i
Pause : DispHome
EndPrgm
```

Méthode de Lagrange

```
lagrange()
Prgm
Local a,b,n,u,e,t,d
ClrIO
Input "borne inferieure de [a,b]",a
Input "borne superieure de [a,b]",b
Input "erreur",e
0 → n
a → u
y1(u-e)*y1(u+e) → t
(b*y1(u)-u*y1(u))/(y1(b)-y1(u)) → d
While t>0
While abs(d) ⩾ e
(b*y1(u)-u*y1(u))/(y1(b)-y1(u)) → d
u-d → u
n+1 → n
EndWhile
y1(u-e)*y1(u+e) → t
EndWhile
Disp "valeur approchee :",round(u)
Disp "nombre iterations :",n
Pause : DispHome
EndPrgm
```

Méthode de Newton

```
newton()
Prgm
Local x0,n,u,e,t,d
ClrIO
Input "x0 dans [a,b] :",x0
Input "erreur",e
0 → n
x0 → u
y1(u-e)*y1(u+e) → t
Define h(x)=d(y1(x),x)
y1(u)/(h(u)) → d
While t>0
While abs(d) ⩾ e
y1(u)/(h-u)) → d
u-d → u
n+1 → n
EndWhile
y1(u-e)*y1(u+e) → t
EndWhile
Disp "valeur approchee :",round(u)
Disp "nombre iterations :",n
Pause : DispHome
EndPrgm
```

Méthode des rectangles gauches

```
rectangl()
Prgm
Local a,b,n,c,h,i
Prompt a,b,n
(b-a)/n → h
a → c
0 → i
While c<b
i+f(c) → i
c+h → c
EndWhile
i*h → i
Disp "Integrale approchee :", approx(i)
Pause : DispHome
EndPrgm
```

Méthode des milieux=tangentes

```
milieux()
Prgm
Local a,b,n,c,h,i
Prompt a,b,n
(b-a)/n → h
a+h/2 → c
0 → i
While c<b
i+f(c) → i
c+h → c
EndWhile
i*h → i
Disp "Integrale approchee :", approx(i)
Pause : DispHome
EndPrgm
```

Méthode des trapèzes

```
trapeze()
Prgm
Local a,b,n,c,h,i
Prompt a,b,n
(b-a)/n → h
a+h → c
(f(a)+f(b))/2 → i
While c<b
i+f(c) → i
c+h → c
EndWhile
i*h → i
Disp "Integrale approchee :", approx(i)
Pause : DispHome
EndPrgm
```

Méthode de Simpson

```
simpson()
Prgm
Local a,b,n,c,h,s
Prompt a,b,n
(b-a)/n → h
a+h/2 → c
0 → s
While c<b
s+f(a)+f(a+h)+4*f(a+h/2) → s
a+h → a
c+h → c
EndWhile
(h/6)*s → i
Disp "Integrale approchee :", approx(i)
Pause : DispHome
EndPrgm
```

Remarques : (i) pour ces programmes, il faut définir la fonction auparavant.
Exemple : (4/(1+(x^2)) → f(x) pour approcher π (en intégrant de 0 à 1).
(ii) pour manipuler les résultats, on peut faire des fonctions (et pas des programmes)