

**derstep**  
Dérivation pas à pas

**Calculatrices :** 89 92 92+ V200

**Niveaux :** 2nde 1ere 1ere S Tale Tale S

**Descriptif :** Groupe de programmes - Permet de calculer pas à pas la dérivée d'une fonction

**Mots-clefs :** dérivée

**Auteur :** Philippe Fortin

**Date de dernière révision :** Septembre 2002

**Présentation :**

Il s'agit d'un groupe dont le programme principal est nommé *derive*, qui permet d'obtenir les différentes étapes du calcul de la dérivée d'une fonction, avec rappel de toutes les formules de calcul utilisées.

**Mode d'emploi :**

Il suffit de vous placer dans le répertoire *derstep*, et de taper *derive()*.

Vous aurez alors à taper l'expression de la fonction à dériver, et à suivre les résultats qui s'affichent à l'écran.

**Sources :**

```
Derive ()  
Prgm  
Local exstring  
"x*cos(x)-5*x^3"→exstring  
ClrHome  
ClrIO  
Dialog  
Title "Derive"  
Text "Calcul pas à pas"  
Text "de la dérivée "  
Text "d(f(x),x)"  
Text ""  
Text "β-version"  
Text "Philippe Fortin"  
Text "ph-fortin@wanadoo.fr"  
EndDlog  
If ok=0:Stop  
Dialog  
Title "Derive"
```

```

Text "Les méthodes qui"
Text "seront utilisées"
Text "sont celles que l'on"
Text "utiliserait à la main..."
EndDlog
If ok=0:Stop
Dialog
Title "Derive"
Text "... sauf lorsqu'une "
Text "auto-simplification"
Text "modifie l'expression"
Text "que l'on souhaite"
Text "dériver."
EndDlog
If ok=0:Stop
Dialog
Title "Derive"
Text "Par exemple,"
Text "(x+1)^5+1"
Text "est automatiquement"
Text "développé par les"
Text "TI-89 & TI-92 PLUS"
EndDlog
If ok=0:Stop
Dialog
Title "Derive"
Text "Utilisez les touches"
Text "de contrôle du curseur"
Text "pour faire défiler les"
Text "résultats qui ne tiennent"
Text "pas dans l'écran"
EndDlog
If ok=0:Stop
Dialog
Title "Derive"
Text "Appuyez ensuite sur [enter]"
Text "pour continuer le calcul"
Text ""
EndDlog
If ok=0:Stop
Local folder
setFold(derstep)→folder
NewProb
Loop
ClrIO
Dialog
Title "Dérivation pas à pas"
Text "Entrez la fonction"
Text "(Appuyez sur ESC pour quitter)"
Text ""
Request "f(x)",exstring
EndDlog
If ok=0:Exit
der("f",expr(exstring),{})
Pause "Fin du calcul pas à pas."
EndLoop

```

```

setFold(#folder)
DispHome
EndPrgm

Tvar (ex)
when(ex=x,true,false,false)

Derku (θf,ex,lex)
Prgm
Local nom,l
when(θf="u", "w", "u")→nom
dispex(θf,k*#nom(x))
dispder(θf,k*d(#nom(x),x))
If dim(lex)>2 Then
augment({"prod"},right(lex,dim(lex)-1))→l
Else
{}→l
EndIf
der(nom,ex/(lex[1]),1)
lex[1]*res→res
dispder(θf,res)
EndPrgm

Dersum (θf,ex,lt)
Prgm
Local i,nbterm,cum,nom,somform,tform
dim(lt)→nbterm
Ø→somform
For i,1,nbterm
θf&string(i)→nom
#nom(x)→tform

If part(lt[i],Ø)="-" Then
somform-tform→somform
Else
somform+tform→somform
EndIf
EndFor

Disp #θf(x)=somform
dispder(θf,d(somform,x))

Ø→cum
For i,1,nbterm
Disp "Dérivée du terme n°"&string(i)
θf&string(i)→nom
If part(lt[i],Ø)="-" Then
der(nom,-lt[i],{}):cum-res→cum
Else
der(nom,lt[i],{}):cum+res→cum
EndIf
EndFor

Disp "La dérivée de"
dispex(θf,somform)
Disp "est donc :"

```

```

cum→res
dispder(θf,res)
EndPrgm

Deropp (θf,ex,lex)
Prgm
Local nom
when(θf="u","w","u")→nom
dispex(θf,-#nom(x))
dispder(θf,-d(#nom(x),x))
der(nom,lex[1],{})
-res→res
Disp "et donc"
dispder(θf,res)
EndPrgm

Dervar (θf,ex,lex)
Prgm
1→res
dispder(θf,res)
EndPrgm

Dispex (fname,ex)
Prgm
Local y
#fname(x)→y
Pause y=ex
EndPrgm

Derstep (ex)
Prgm
NewProb
ClrIO
der("f",ex,{})
Pause "fin du calcul."
DispHome
EndPrgm

Derfond (θf,ex,lex)
Prgm
Disp "La dérivée de "&θf&" est"
d(ex,x)→res
dispder(θf,res)
EndPrgm

Typeaf (ex)
Func
when(d(ex,x,2)=∅,true,false,false)
EndFunc

Dispder (fname,dex)
Prgm
Local y
#fname(x)→y
Pause d(y,x)=dex
EndPrgm

```

```

Derprod (θf,ex,lf)
Prgm
Local i,nbfact,cum,nom,prodform,tform
dim(lf)→nbfact
1→prodform
For i,1,nbfact
  θf&string(i)→nom
  #nom(x)→tform
  prodform*tform→prodform
EndFor
Disp "#θf(x)=prodform
dispder(θf,d(prodform,x))
For i,1,nbfact
  Disp "Dérivée du facteur n°"&string(i)
  θf&string(i)→nom
  der(nom,lf[i],{})
EndFor
Disp "La dérivée de"
dispex(θf,prodform)
Disp "est donc :"
d(ex,x)→res
dispder(θf,res)
EndPrgm

Derquot (θf,ex,lex)
Prgm
Local num,den,ex1,ex2,dex1,dex2
lex[1]→ex1
lex[2]→ex2
If θf="u" or θf="v" Then
  "num"&θf→num
  "den"&θf→den
Else
  "u"→num
  "v"→den
EndIf
If when(d(ex1,x)=∅,true,false,false) Then
  © f=λ/d avec λ constant
  dispex(θf,ex1/(#den(x)))
  der(den,ex2,{}):res→dex2
  Disp "(k/v)'=-k v'/v²"
  Disp "On a donc"
  -ex1*dex2/ex2^2→res
  dispder(θf,res)

Else
  © f=n/d
  dispex(θf,#num(x)/(#den(x)))
  der(num,ex1,{}):res→dex1
  der(den,ex2,{}):res→dex2
  Disp "(u/v)'=(u'v-uv')/v²"
  Disp "On a donc"
  (dex1*ex2-ex1*dex2)/ex2^2→res
  dispder(θf,res)
EndIf

```

```

EndPrgm

Derpuiss (θf,ex,lex)
Prgm
Local s,ex1,ex2,dex1,dex2,nom
lex[1]→ex1
lex[2]→ex2
when(θf="u","w","u")→nom

If when(ex1=x,true,false,false) Then
  © f(x)=x^n
  Disp "(x^n)' = n x^(n-1)"
  ex2*ex1^(ex2-1)→res
  dispder(θf,res)
Else
  © f(x)=u(x)^n
  dispex(θf,(#nom(x))^ex2)
  Disp "(u^n)' = n u' u^(n-1)"
  dispex(nom,ex1)
  der(nom,ex1,{}) : res→dex1
  ex2*dex1*ex1^(ex2-1)→res
  dispder(θf,ex2*d(#nom(x),x)*(#nom(x))^(ex2-1))
  dispder(θf,res)
EndIf
EndPrgm

Lfact (ex)
Func
Local l1,l2,l
If part(ex,0)="*" Then
Lfact(part(ex,1))→l1
Lfact(part(ex,2))→l2
{l1[1]*l2[1]}→l
augment(l,right(l1,dim(l1)-1))→l
augment(l,right(l2,dim(l2)-1))→l
l
Else
when(d(ex,x)=∅,{ex},{1,ex},{1,ex})
EndIf
EndFunc

Der (θf,ex,lex)
Prgm
Local nom
If lex={} Then
  der(θf,ex,struct(ex))
Else
  If lex[1]!="fonc"
    Disp "Calcul de la dérivée de"
    dispex(θf,ex)
    "der"&lex[1]→nom
    #nom(θf,ex,right(lex,dim(lex)-1))
EndIf
EndPrgm

```

```

Folder
"derstep"

Lterm (ex)
Func
Local op
part(ex, $\emptyset$ ) $\rightarrow$ op
If op= "+" Then
augment(lterm(part(ex,1)),lterm(part(ex,2)))
ElseIf op= "-" Then
augment(lterm(part(ex,1)),lterm(~part(ex,2)))
Else
{ex}
EndIf
EndFunc

Deraf ( $\theta$ f,ex,lex)
Prgm
d(ex,x) $\rightarrow$ res
Disp "la dérivée de ax+b est a"
dispder( $\theta$ f,res)
EndPrgm

Struct (ex)
Func
Local op,d1,d2,lf,u,v
d(ex,x) $\rightarrow$ d1
d(d1,x) $\rightarrow$ d2
part(ex, $\emptyset$ ) $\rightarrow$ op
If when(d1= $\emptyset$ ,true,false,false) Then
{"cons",ex}
ElseIf when(d2= $\emptyset$ ,true,false,false) Then
when(ex=x,{"var",ex}, {"af",ex}, {"af",ex})
ElseIf op= "+" or op= "-" Then
augment({ "sum"},lterm(ex))
ElseIf op= "*" Then
lfact(ex) $\rightarrow$ lf
If when(lf[1]-1= $\emptyset$ ,true,false,false) Then
augment({ "prod"},right(lf,dim(lf)-1))
Else
augment({ "ku"},lf)
EndIf
ElseIf op= "/" Then
{"quot",part(ex,1),part(ex,2)}
ElseIf op= "^" Then
part(ex,1) $\rightarrow$ u
part(ex,2) $\rightarrow$ v
If when(d(v,x)= $\emptyset$ ,true,false,false) Then
{"puiss",u,v}
ElseIf when(ex= $e^x$ ,true,false,false) Then
{"fonc"," $e^x$ ",x}
Else
{"comp"," $e^x$ ",v*ln(u)}
EndIf
ElseIf op= "-" Then
{"opp",part(ex,1)}

```

```

ElseIf part(ex)=1 Then
part(ex,1)→u
If when(u=x,true,false,false) Then
 {"fonc",part(ex,∅),part(ex,1)}
Else
 {"comp",part(ex,∅),part(ex,1)}
EndIf
Else
 {"?"}
EndIf
EndFunc

Dercomp (θf,ex,lex)
Prgm
Local op,arg,darg,dφ,y,derform,exform,nom
lex[1]→op
lex[2]→arg
when(θf="u","w","u")→nom

If op≠"e^" Then
Pause "C'est une fonction composée"
expr(op&"(x)")→y
Disp #θf(x)=φ(#nom(x))
Disp φ(x)=#op(x)
dispex(nom,arg)
Disp "la dérivée de φ(u(x)) est"
Disp "u'(x).φ'(u(x))"
y|x=θθθ→exform
expr("exform|θθθ=&nom&"(x))→exform

der("φ",y,{ "fonc",op,x}):res→dφ

dφ|x=θθθ→derform
expr("derform|θθθ=&nom&"(x))→derform
d(#nom(x),x)*derform→derform
dispex(θf,exform)
dispder(θf,derform)

der(nom,arg,struct(arg))
res→darg

dφ|x=θθθ→res
res|θθθ=arg→res

darg*res→res
Disp "et donc"
dispder(θf,res)
Else
© cas de e^(w(x))
dispex(θf,e^(#nom(x)))
dispex(nom,arg)
dispder(θf,d(e^(#nom(x)),x))
dispder(θf,d(#nom(x),x)*#θf(x))
der(nom,arg,{})
Disp "et donc"
dispder(θf,d(#nom(x),x)*#θf(x))

```

```
res*ex→res
dispder(θf,res)
EndIf
EndPrgm

Tcons (ex)
when( $d(ex,x)=\emptyset$ ,true,false,false)

Dercons (θf,ex,lex)
Prgm
Disp θf&" est constante"
∅→res
dispder(θf,res)
EndPrgm
```