Algorithmes en analyse

Illustrations avec XCAS et CAML

Guillaume CONNAN

IREM de Nantes

11 janvier 2010

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

irem

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Ayez toujours en tête qu'un ordinateur ne peut pas travailler avec des réels !

Par essence, il a un nombre fini de bits sur lesquels il peut coder des nombres...

Ayez toujours en tête qu'un ordinateur ne peut pas travailler avec des réels !
Par essence, il a un nombre fini de bits sur lesquels il peut coder des nombres...

Les nombres *flottants* qu'il manipule sont donc des classes qui représentent chacune une infinité de réels (mais tout en constituant un intervalle cohérent).

Si on n’y fait pas attention, cela peut créer des surprises...

Entrons trois nombres sur XCAS :

```
x:=10;
y:=-5;
z:=5.00000001;
```

Alors :

```
(x*y)+(x*z)
```

renvoie :

```
1.000000012e-07
```

Mais

```
x*(y+z)
```

renvoie :

```
9.999999939e-08
```

Alors :

```
(x*y)+(x*z)
```

renvoie :

```
1.000000012e-07
```

Mais

```
x*(y+z)
```

renvoie :

```
9.999999939e-08
```

Le problème est le même avec CAML :

```
# let x,y,z=10.,-5.,5.00000001;;
val x : float = 10.
val y : float = -5.
val z : float = 5.00000001

# (x*.y)+.(x*.z);;
- : float = 1.00000001168609742e-07

# x*.(y+.z);;
- : float = 9.9999993922529e-08
```

Ainsi l'addition n'est pas toujours distributive sur la multiplication quand on travaille sur les flottants !

Ces petites erreurs qui s'accumulent peuvent en créer de grandes !

Ainsi l'addition n'est pas toujours distributive sur la multiplication quand on travaille sur les flottants !
Ces petites erreurs qui s'accumulent peuvent en créer de grandes !

Créons une fonction qui renvoie l'approximation de la dérivée d'une fonction en un nombre donné avec un « dx » valant 10^{-10} :

```
# let dernum(f)=function
          x->(f(x+.0.0000000001)-.f(x))/.0.0000000001;;
```

Dérivons le sinus cinq fois de suite et évaluons la fonction obtenue en 0 :

```
# dernum(dernum(dernum(dernum(dernum(sin)))))(0.);;
- : float = -1.33226762955018773e+25
```

```
# let dernum(f)=function
          x->(f(x+.0.0000000001)-.f(x))/.0.0000000001;;
```

Dérivons le sinus cinq fois de suite et évaluons la fonction obtenue en 0 :

```
# dernum(dernum(dernum(dernum(dernum(sin)))))(0.);;
- : float = -1.33226762955018773e+25
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

On a créé une liste de coordonnées du type $(x, f(x))$ avec un pas de h entre a et b.

```
liste_points(f,a,b,h):={
if(a>b)then{[b,f(b)]}
else{[a,f(a)],liste_points(f,a+h,b,h)}
}:;
```

On a créé une liste de coordonnées du type $(x, f(x))$ avec un pas de h entre a et b.

```
liste_points(f,a,b,h):={
if(a>b)then{[b,f(b)]}
else{[a,f(a)],liste_points(f,a+h,b,h)}
}:;
```

On relie ces points par des segments avec `polygonplot(liste de coordonnées)` :

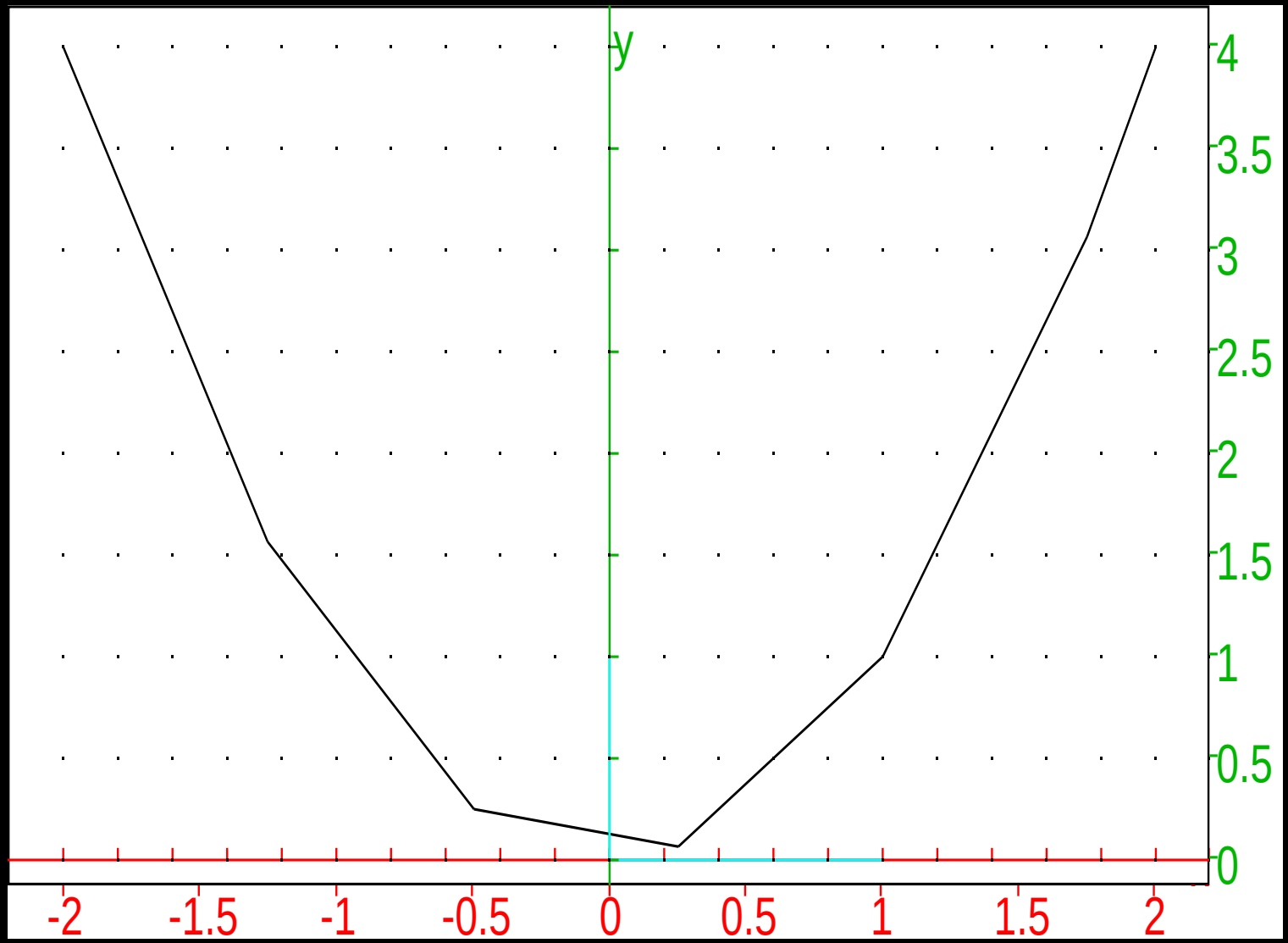
```
graph(f,a,b,h):={
polygonplot([liste_points(f,a,b,h)])
}:;
```

On relie ces points par des segments avec `polygonplot(liste de coordonnées)` :

```
graph(f,a,b,h):={
polygonplot([liste_points(f,a,b,h)])
}:;
```

Par exemple, on trace la fonction carrée entre -2 et 2 avec un pas de $0,75$:

```
graph(x->x^2,-2,2,0.75)
```


y
4
3.5
3
2.5
2
1.5
1
0.5
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

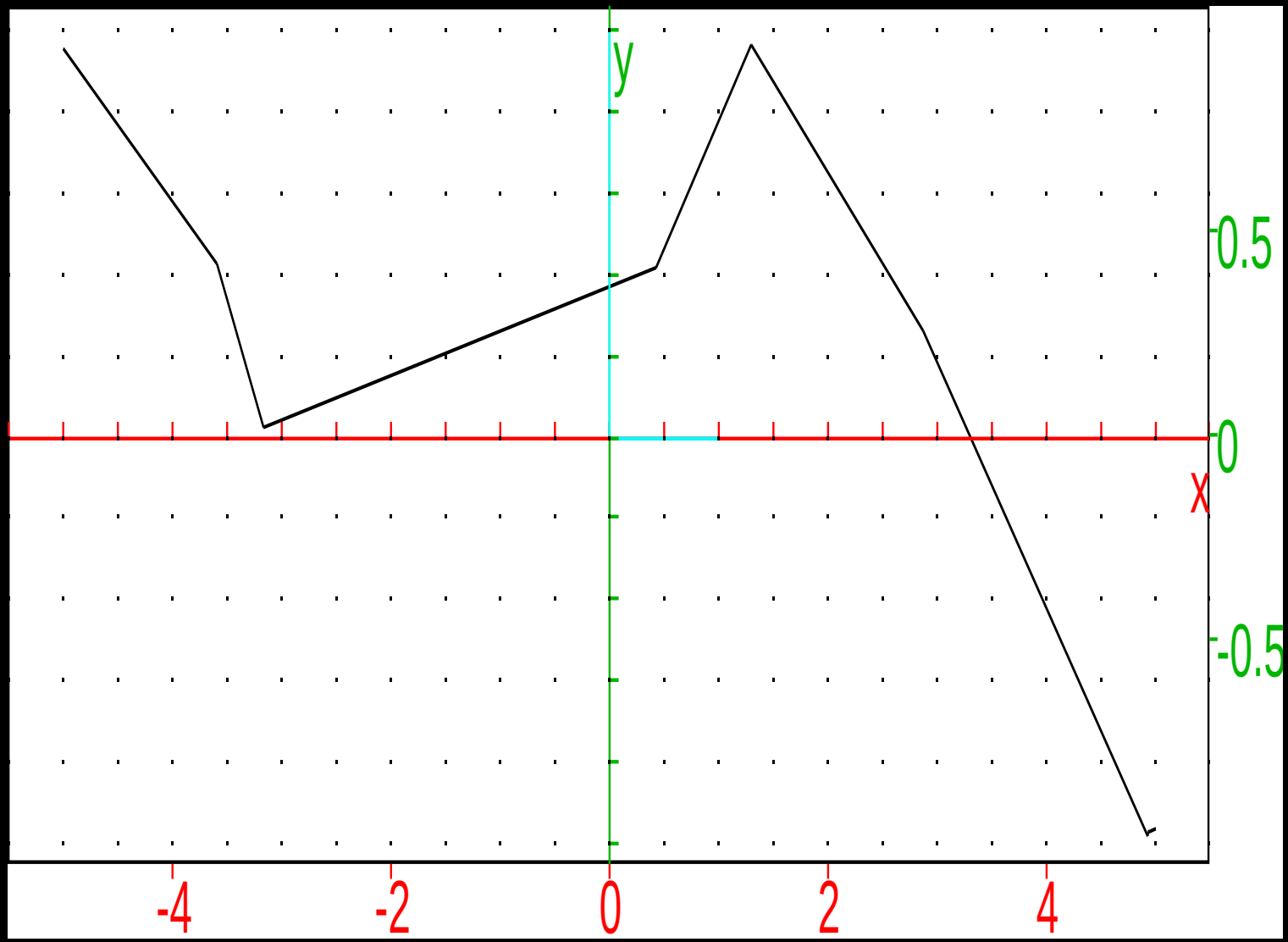
On interroge au hasard un réel et on lui demande son image par une certaine fonction : cela nous permettra-t-il d'avoir une bonne idée de l'allure de la courbe ?

```
liste_points_hasard(f,a,a0,b,n):={
if(n==0)then{[b,f(b)]}
else{[a0,f(a0)],liste_points_hasard(f,a,a+rand(0,1)*(b-a),b
    ,n-1)}
}:;
```

```
graph_hasard(f,a,b,n):={
polygonplot([liste_points_hasard(f,a,a,b,n)])
}:;
```

Alors on obtient pour la fonction sinus entre -5 et 5 avec 8 « sondés » :

```
graph_hasard(x->sin(x),-5,5,8)
```


y
x
0.5
0
-0.5
-4
-2
0
2
4

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Comment faire dessiner une courbe en dents de scie, chaque dent ayant une largeur de 2 et une hauteur de 1 ? Peut-on avoir une formule explicite qui permette de calculer l'image de n'importe quel nombre ?

Les coefficients directeurs valent alternativement 1 et -1.

Les ordonnées s'annulent aux points d'abscisses paires...

Comment faire dessiner une courbe en dents de scie, chaque dent ayant une largeur de 2 et une hauteur de 1 ? Peut-on avoir une formule explicite qui permette de calculer l'image de n'importe quel nombre ?
Les coefficients directeurs valent alternativement 1 et -1.
Les ordonnées s'annulent aux points d'abscisses paires...

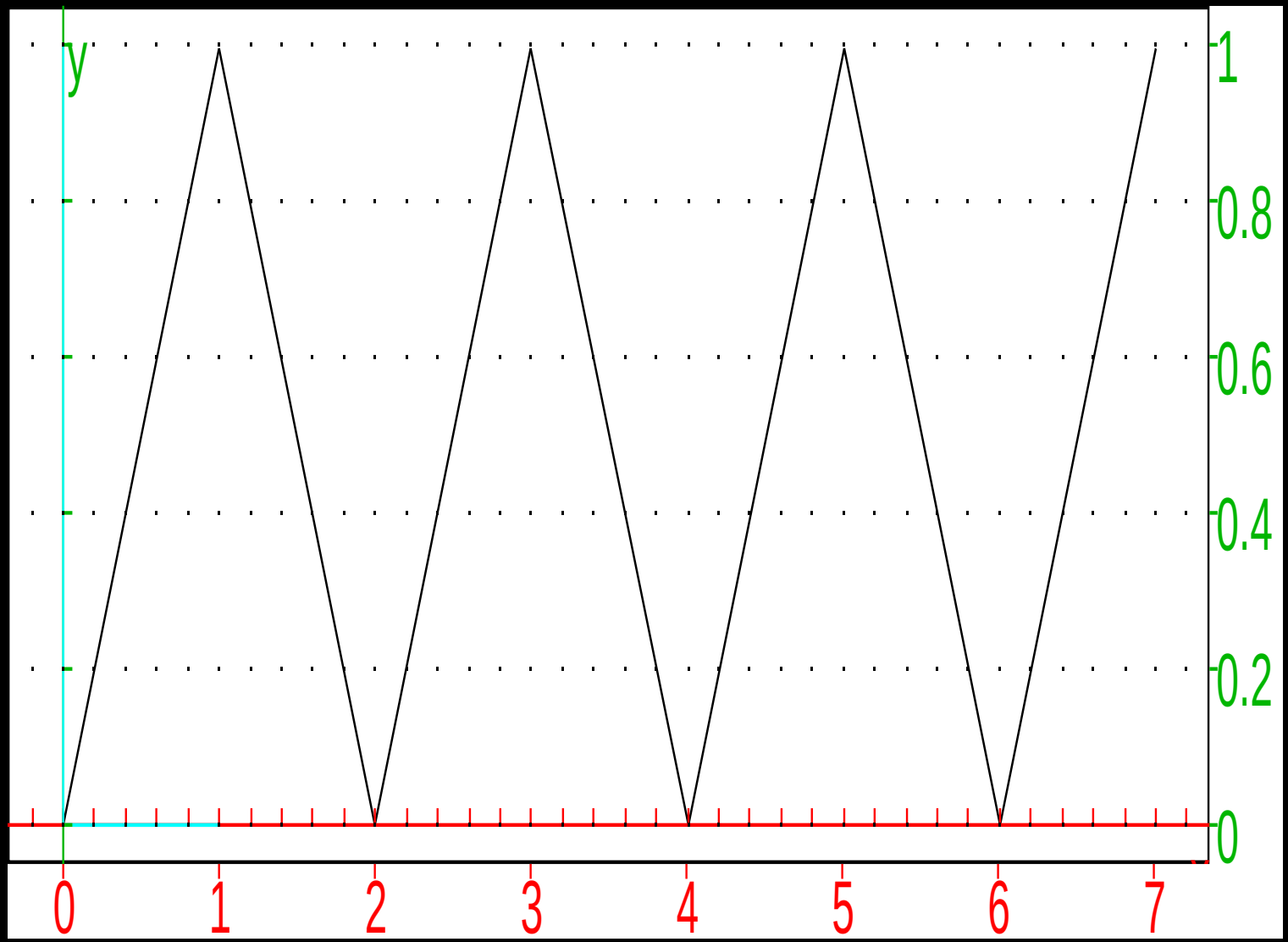
Comment faire dessiner une courbe en dents de scie, chaque dent ayant une largeur de 2 et une hauteur de 1 ? Peut-on avoir une formule explicite qui permette de calculer l'image de n'importe quel nombre ?
Les coefficients directeurs valent alternativement 1 et -1.
Les ordonnées s'annulent aux points d'abscisses paires...

```
scie_rec(n):={
if(n==0)then{[0,0]}
  else{[n,(1-(-1)^n)/2],scie_rec(n-1)}
}:;
```

```
scie(n):={
polygonplot([scie_rec(n)])
}:;
```

```
scie_rec(n):={
if(n==0)then{[0,0]}
  else{[n,(1-(-1)^n)/2],scie_rec(n-1)}
}:;
```

```
scie(n):={
polygonplot([scie_rec(n)])
}:;
```


y
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7

Le tracé a été obtenu sans formule explicite pour chaque réel. Cherchons une telle formulation :

```
f(x):={
   if(irem(floor(x),2)==0)
      then{x-floor(x)}
      else{1-x+floor(x)}
}:;
```

sachant que `floor` donne la partie entière et `irem` le reste de la division euclidienne.

Le tracé a été obtenu sans formule explicite pour chaque réel. Cherchons une telle formulation :

```
f(x):={
   if(irem(floor(x),2)==0)
      then{x-floor(x)}
      else{1-x+floor(x)}
}:;
```

sachant que `floor` donne la partie entière et `irem` le reste de la division euclidienne.

Le tracé a été obtenu sans formule explicite pour chaque réel. Cherchons une telle formulation :

```
f(x):={
   if(irem(floor(x),2)==0)
      then{x-floor(x)}
      else{1-x+floor(x)}
}:;
```

sachant que `floor` donne la partie entière et `irem` le reste de la division euclidienne.

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

On repère graphiquement qu'une fonction semble atteindre par exemple un minimum. Pour obtenir une approximation de ce minimum, on fournit un point de départ et une précision.

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

En CAML :

```
# let rec min_rec(f,xo,h)=
    if f(xo+.h)>f(xo) then xo
    else min_rec(f,xo+.h,h);;
```

En XCAS :

```
min_rec(f,xo,h):={
if(f(xo+h)>f(xo)){return(xo)}
else{min_rec(f,xo+h,h)}
}
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Entrées : fonction f, point de départ xo et incrément i
Initialisation : Y⟵ f(xo) X⟵ xo+i
début
  tant que *t(X)<Y* **faire**
    Y⟵ evalf(t(X)) X⟵ X+i
  retourner *X-i*
fin

```
minimum(f,xo,p):={
local X,Y,compteur;
Y:=f(xo);
X:=xo+10^(-p);
compteur:=0 // on rajoute un compteur pour que la
    compilation ne dure pas trop longtemps
tantque evalf(f(X))<Y et compteur<10^p faire
    Y:=evalf(f(X));
    X:=X+10^(-p);
    compteur:=compteur+1;
ftantque:;
 X-10^(-p);
}:;
```

Par exemple, pour la fonction $x \mapsto \sqrt{x^2+25} + \sqrt{(x-18)^2+49}$, il semble que le minimum soit vers 7.

Demandons une approximation du minimum à 10^{-5} près :

```
minimum(x->sqrt((x)^2+25)+sqrt((x-18)^2+49),7.0,5)
```

et on obtient 7.50000

Par exemple, pour la fonction $x \mapsto \sqrt{x^2+25} + \sqrt{(x-18)^2+49}$, il semble que le minimum soit vers 7.
Demandons une approximation du minimum à 10^{-5} près :

```
minimum(x->sqrt((x)^2+25)+sqrt((x-18)^2+49),7.0,5)
```

et on obtient 7.50000

Par exemple, pour la fonction $x \mapsto \sqrt{x^2+25} + \sqrt{(x-18)^2+49}$, il semble que le minimum soit vers 7.
Demandons une approximation du minimum à 10^{-5} près :

```
minimum(x->sqrt((x)^2+25)+sqrt((x-18)^2+49),7.0,5)
```

et on obtient 7.50000

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

On peut améliorer la situation en affinant petit à petit le balayage : d'abord h puis $h/10$, etc.

Avec CAML :

```
# let rec min_bis(f,xo,h,p)=
  if abs_float(h)<p then xo
  else if f(xo+.h)>f(xo)
       then min_bis(f,xo,-.0.1*.h,p)
       else min_bis(f,xo+.h,h,p);;
```

Avec XCAS :

```
min_bis(f,xo,h,p):={
if(abs(h)<p)then{xo}
else{if(f(xo+h)>f(xo))
     then{min_bis(f,xo,-0.1*h,p)}
     else{min_bis(f,xo+h,h,p)}
}
}:;
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Pour résoudre une équation du type $f(x) = 0$, on recherche graphiquement un intervalle $[a, b]$ où la fonction semble changer de signe.
On note ensuite m le milieu du segment $[a, b]$. On évalue le signe de $f(m)$. Si c'est le même que celui de $f(a)$ on remplace a par m et on recommence. Sinon, c'est b qu'on remplace et on recommence jusqu'à obtenir la précision voulue.

Pour résoudre une équation du type $f(x) = 0$, on recherche graphiquement un intervalle $[a, b]$ où la fonction semble changer de signe. On note ensuite m le milieu du segment $[a, b]$. On évalue le signe de $f(m)$. Si c'est le même que celui de $f(a)$ on remplace a par m et on recommence. Sinon, c'est b qu'on remplace et on recommence jusqu'à obtenir la précision voulue.

Pour résoudre une équation du type $f(x) = 0$, on recherche graphiquement un intervalle $[a, b]$ où la fonction semble changer de signe. On note ensuite m le milieu du segment $[a, b]$. On évalue le signe de $f(m)$. Si c'est le même que celui de $f(a)$ on remplace a par m et on recommence. Sinon, c'est b qu'on remplace et on recommence jusqu'à obtenir la précision voulue.

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

En CAML :

```
# let rec dicho_rec(f,a,b,eps)=
    if abs_float(b-.a)<eps then 0.5*.(b+.a)
    else if f(a)*.f(0.5*.(b+.a))>0. then dicho_rec(f,0.5*.(
        b+.a),b,eps)
          else  dicho_rec(f,a,0.5*.(b+.a),eps);;
```

ce qui donne :

```
# dicho_rec( (fun x->(x*.x-.2.)),1.,2.,0.00001);;
- : float = 1.4142112731933594
```

En CAML :

```
# let rec dicho_rec(f,a,b,eps)=
    if abs_float(b-.a)<eps then 0.5*.(b+.a)
    else if f(a)*.f(0.5*.(b+.a))>0. then dicho_rec(f,0.5*.(
        b+.a),b,eps)
         else  dicho_rec(f,a,0.5*.(b+.a),eps);;
```

ce qui donne :

```
# dicho_rec( (fun x->(x*.x-.2.)),1.,2.,0.00001);;
- : float = 1.4142112731933594
```

En XCAS :

```
dicho_rec(f,a,b,eps):={
if(evalf(b-a)<eps){return(0.5*(b+a))};
if(f(a)*f(0.5*(b+a))>0){return(dicho(f,0.5*(b+a),b,eps))}
                  else{return(dicho(f,a,0.5*(b+a),eps))}
};:;
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Entrées : une fonction f, les bornes a et b, une précision p
Initialisation : aa⟵ a, bb⟵ b
début
  tant que *bb-aa>p* **faire**
    si *signe de f((aa+bb)/2) est le même que celui de f(bb)* **alors**
      bb⟵ (aa+bb)/2
    sinon
      aa⟵ (aa+bb)/2
  retourner *(aa+bb)/2*
fin

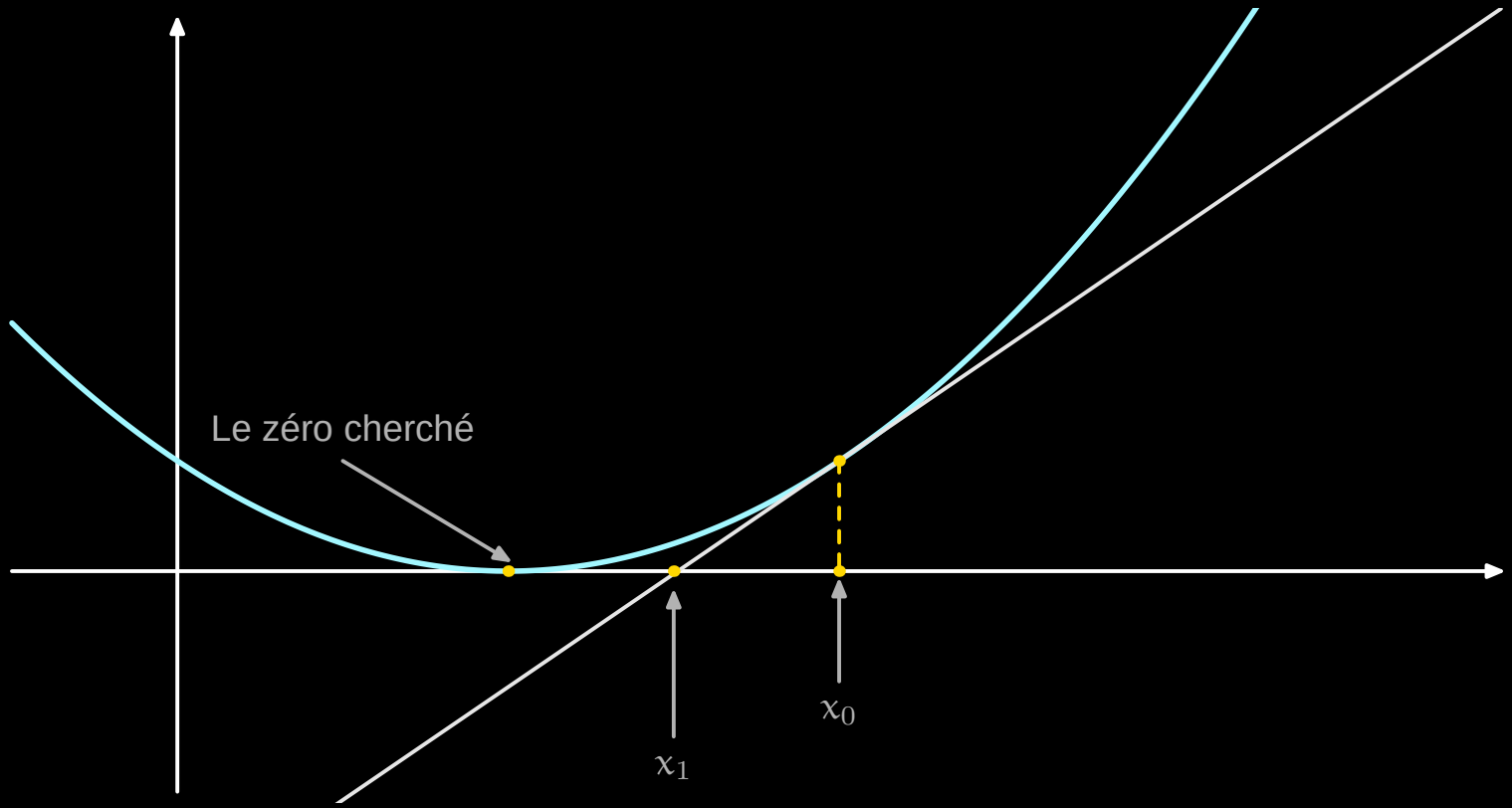
Avec XCAS, il faut juste ne pas oublier de régler quelques problèmes de précision :

```
dicho(F,p,a,b):={
local aa,bb,k,f;
aa:=a;
bb:=b;
epsilon:=1e-100; // on règle le "zéro" de XCAS à une valeur
     suffisamment petite
f:=unapply(F,x);// on transforme l'expression f en fonction
k:=0; // on place un compteur
 tantque evalf(bb-aa,p)>10^(-p) faire
    si sign(evalf(f((bb+aa)/2),p))==sign(evalf(f(bb),p))
      alors bb:=evalf((aa+bb)/2,p);
      sinon aa:=evalf((aa+bb)/2,p);
      k:=k+1; // un tour de plus au compteur
    fsi;
 ftantque;
return evalf((bb+aa)/2,p)+" est la solution trouvée après "
     +k+ " itérations";
}:;
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Les prérequis sont plus nombreux ! Il faut avoir étudié les suites définies par une relation du type $u_{n+1} = f(u_n)$ et la dérivation... et ne pas être trop perdu en Analyse...
Un exemple de TP possible en terminale S est présenté dans le poly.


Le zéro cherché
x_1
x_0

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

En XCAS : on notera que `function_diff(f)` renvoie la *fonction* dérivée donc `function_diff(f)(x0)` désigne le *nombre* dérivé de f en x_0.

```
newton_rec(f,x0,eps):={
if(evalf(abs(f(x0)/function_diff(f)(x0)))<eps){evalf(x0)}
else{newton_rec(f,evalf(x0-f(x0)/function_diff(f)(x0)),eps)
    }
};;
```

En XCAS : on notera que `function_diff(f)` renvoie la *fonction* dérivée donc `function_diff(f)(x0)` désigne le *nombre* dérivé de f en x_0.

```
newton_rec(f,x0,eps):={
if(evalf(abs(f(x0)/function_diff(f)(x0)))<eps){evalf(x0)}
else{newton_rec(f,evalf(x0-f(x0)/function_diff(f)(x0)),eps)
    }
}:;
```

Alors, après avoir fixé par exemple la précision à 100 :

```
DIGITS:=100:;
newton_rec(x->x^2-2,1.0,evalf(10^(-99)))
```

On obtient immédiatement :

1.41421356237309504880168872420969807856967187537694807317667973

Alors, après avoir fixé par exemple la précision à 100 :

```
DIGITS:=100:;
newton_rec(x->x^2-2,1.0,evalf(10^(-99)))
```

On obtient immédiatement :

1.41421356237309504880168872420969807856967187537694807317667973

En OCAML : il y a un petit plus car il faut définir une dérivée approchée.

```
# let der(f,x,dx)=(f(x+.dx)-.f(x))/.dx;;
```

Maintenant la partie Newton :

```
# let rec newton_rec(f,xo,dx,eps)=
    if abs_float(f(xo)/.der(f,xo,dx))<eps then xo
    else newton_rec(f,xo-.f(xo)/.der(f,xo,dx),dx,eps);;
```

```
# let k(x)=x*.x-.2.;;

# newton_rec(k,1.,0.0001,0.000000001);;
- : float = 1.4142135624530596
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Ça se complique un peu : évidemment, car il faut faire de l'informatique au lieu de se concentrer sur les mathématiques... :-)

Entrées : fonction f précision p premier terme $a0$ nombre maximum d'itération `// Pour éviter de « planter » si on tombe sur un cas pathologique`

Initialisation : $fp \longleftarrow$ dérivée de f ;

compteur$\longleftarrow$ 0 `// le compteur d'itérations`

$un \longleftarrow u0 - \frac{f(u0)}{fp(u0)}$

début

  tant que $\left|\frac{f(xn)}{fp(xn)}\right|$ *est plus grand que la précision p et que* $k < N$ **faire**

    si $fp(un) = 0$ **alors**

      On sort de la boucle avant de diviser par zéro et on explique pourquoi

    $un \longleftarrow un - \frac{f(un)}{fp(un)}$ compteur$\longleftarrow$ compteur+1

fin

retourner *L'approximation et la valeur du compteur*

Comparez ensuite avec les résultats trouvés avec la dichotomie.
Regardez également ce qui se passe avec l'équation $(x-1)^4 = 0$: quelle précaution supplémentaire faut-il prendre ? (La preuve n'est évidemment pas envisageable au Lycée).

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Supposons qu'on connaisse $f'(x)$ en fonction de x et de $f(x)$ sous la forme $f'(x) = u\big(f(x), x\big)$.

On utilise le fait que $f'(x) \approx \frac{f(x+h)-f(x)}{h}$. Alors

$$f(x+h) \approx h \cdot u\big(f(x), x\big) + f(x)$$

La traduction algorithmique est alors directe.

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Pour la liste des coordonnées de points :

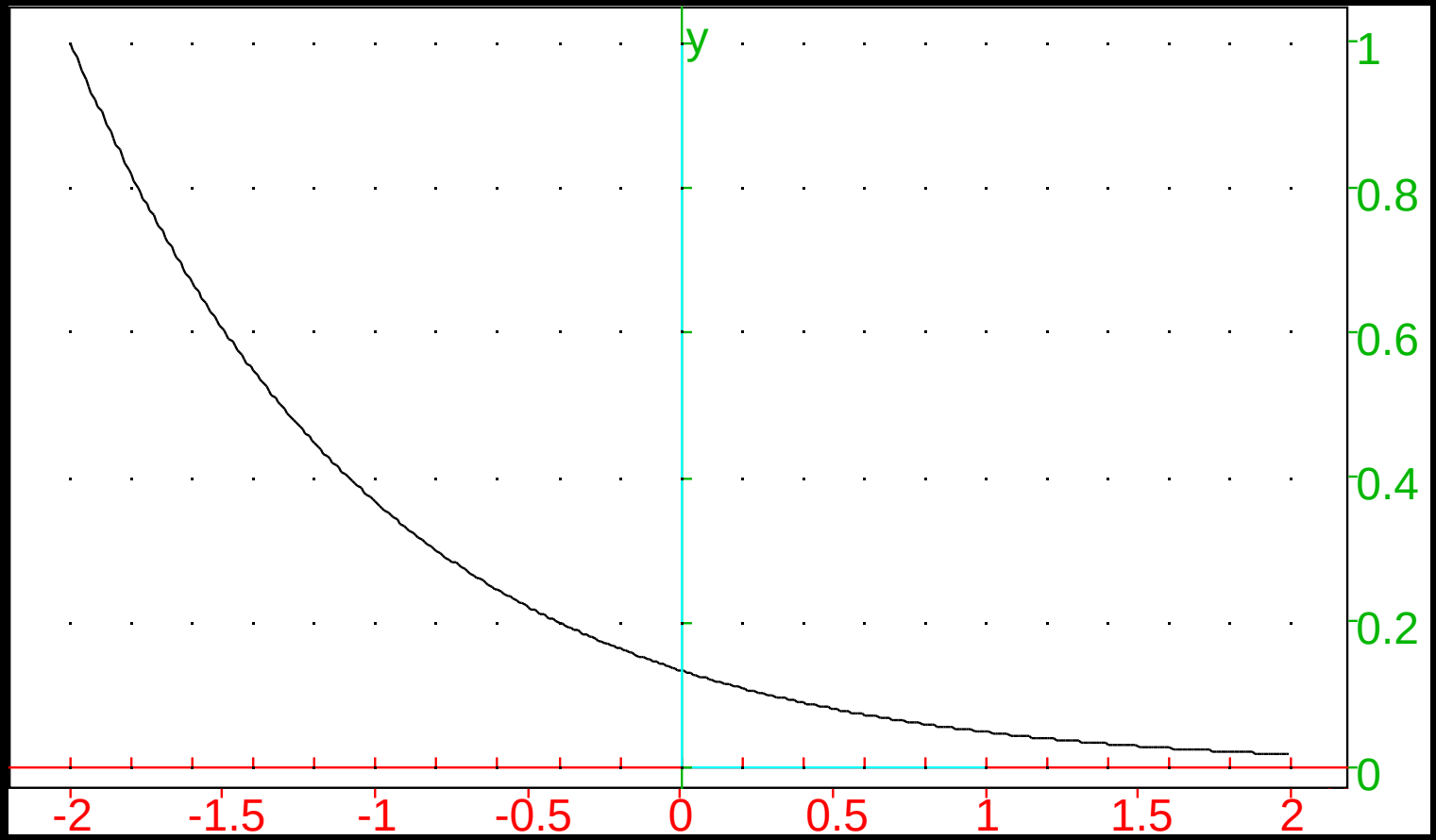
```
Euler(u,x,xmax,y,h,liste):={
 if(x>=xmax)
   then{liste}
   else{ Euler(u,x+h,xmax,y+u(y,x)*h,h,[op(liste),[x,y]])}
}:;
```

Puis pour le tracé :

```
trace_euler(u,xo,xmax,yo,h):={
polygonplot([Euler(u,xo,xmax,yo,h,[])])
}:;
```

Alors, pour résoudre graphiquement $y' = -y$ avec $y(0) = 1$, sur $[-2; 2]$ avec un pas de $0,01$ on entre :

```
trace_euler((y,x)->-y,-2,2,1,0.01)
```


y
1
0.8
0.6
0.4
0.2
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

On effectue une division régulière en découpant notre intervalle d'étude $[a; b]$ en N points.

```
EulerImp(u,N,a,b,yo):={
S:=NULL;
X:=a;
Y:=yo;
h:=(b-a)/N
pour k de 0 jusque N pas 1 faire
X:=a+k*h;
Y:=Y+u(Y,X)*h;
S:=S,[X,Y];
fpour
polygonplot([S]);
}:;
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

```
# let rec integ(f,a,b,dx)=
      if a>b then 0.
      else f(a)*.dx+.integ(f,a+.dx,b,dx);;
```

Par exemple :

```
# let g(x)=x;;

# integ(g,0.,1.,0.0001);;
```

donne :

```
- : float = 0.50004999999960249
```

C'est-à-dire $\int_0^1 x \mathrm{d}\, x \approx 0.5$

Par exemple :

```
# let g(x)=x;;

# integ(g,0.,1.,0.0001);;
```

donne :

```
- : float = 0.50004999999960249
```

C'est-à-dire $\int_0^1 x \mathrm{d}\, x \approx 0.5$

Par exemple :

```
# let g(x)=x;;

# integ(g,0.,1.,0.0001);;
```

donne :

```
- : float = 0.50004999999960249
```

C'est-à-dire $\int_0^1 x \mathrm{d}\, x \approx 0.5$

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

```
integ_rectangle(f,a,b,dx):={
local aa,I;
aa:=a;
I:=0;
 while(aa<b){
 I:=I+f(aa)*dx;
 aa:=aa+dx;
 }
return(I)
}:;
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

```
# let rec integ_trap(f,a,b,dx)=
      if a>b then 0.
      else 0.5*.dx*.(f(a)+.f(a+.dx))+.integ_trap(f,a+.dx,b,
          dx);;
```

```
# integ_trap(g,0.,1.,0.0001);;
- : float = 0.500100004999960213
```

```
# let rec integ_trap(f,a,b,dx)=
      if a>b then 0.
      else 0.5*.dx*.(f(a)+.f(a+.dx))+.integ_trap(f,a+.dx,b,
          dx);;
```

```
# integ_trap(g,0.,1.,0.0001);;
- : float = 0.50010000499960213
```

Sommaire

1. Les réels et le processeur
2. Tracé de représentations graphiques
   - Version récursive à pas constant
   - Version récursive par sondage aléatoire
3. Fonctions définies par morceaux
4. Recherche de minimum
   - Version récursive
   - Version impérative
   - Avec affinage progressif du pas
5. Dichotomie
   - Version récursive
   - Version impérative
6. Méthode de NEWTON
   - Algorithme récursif
   - Algorithme impératif
7. Méthode d'EULER
   - Version récursive
   - Version impérative
8. Intégration numérique
   - Méthode des rectangles
     - Version récursive
     - Version impérative
   - Méthode des trapèzes
     - Version récursive
     - Version impérative

```
integ_trap(f,a,b,dx):={
local aa,I;
aa:=a;
I:=0;
 while(aa<b){
 I:=I+(f(aa)+f(aa+dx))*dx*0.5;
 aa:=aa+dx;
 }
return(I)
}:;
```