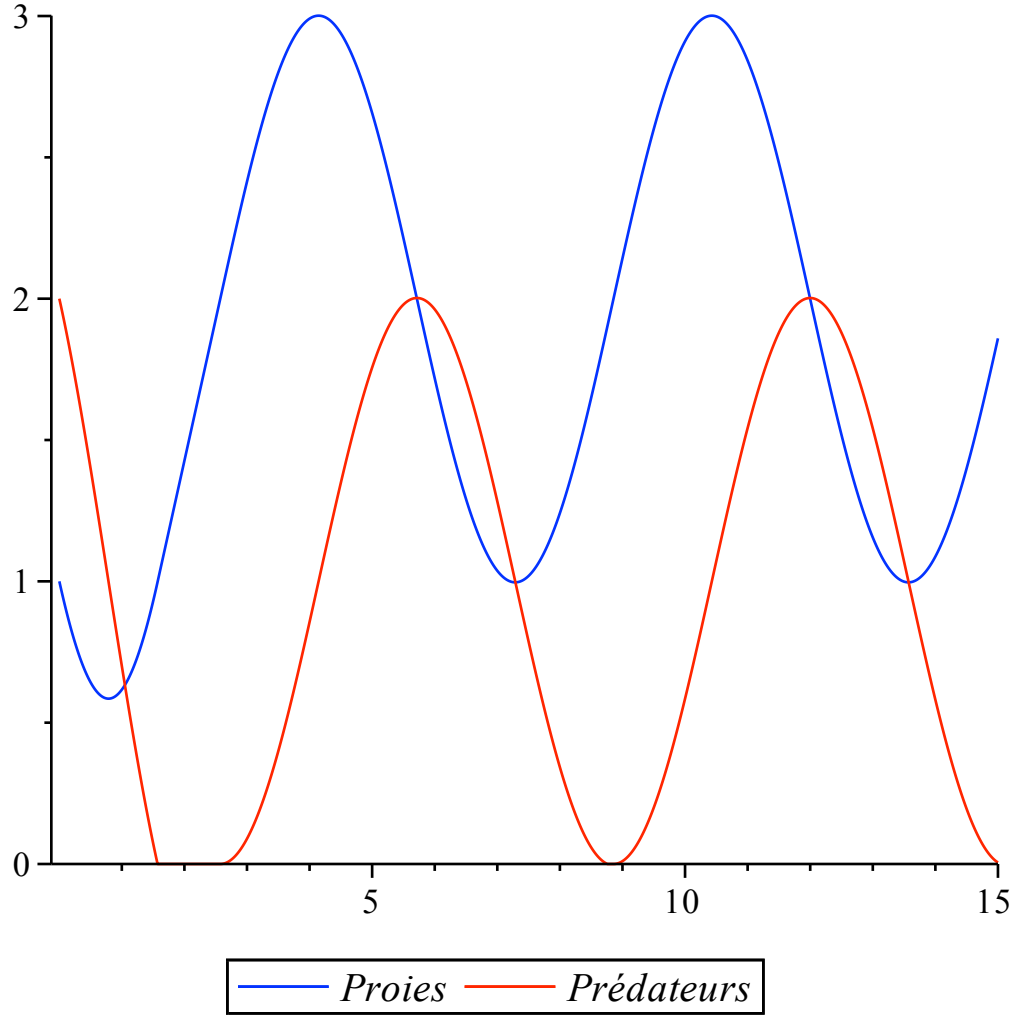
5.7 Exercices

Exercice 32

```
> restart;
> lotka := proc(x0, y0, t0, T, N)
  local h, alpha, gam, delta, beta, n :
   global x, y :
   x := array(1..N) : y := array(1..N) :
   x[1] := x0 : y[1] := y0 :
   alpha := 1. : beta := 1. : delta := 1. : gam := 2. :
   h := evalf((T - t0)/N) :
   for n from 1 to N - 1 do
   x[n + 1] := x[n] + h·(alpha - beta·y[n]) :
   y[n + 1] := y[n] - h·(gam - delta·x[n]) :
   if (x[n + 1] < 0) then x[n + 1] := 0 : fi:
  if (y[n + 1] < 0) then y[n + 1] := 0 : fi:
   od:
   end proc:
> N := 10000 : t0 := 0 : T := 15 :
   lotka(1, 2, t0, T, N) :
> with(plots) :
> G1 := plot([seq(t0 + n·(T - t0)/N, n = 1..N)], [seq(x[n], n = 1..N)], color
      = blue, legend = Proies);
   G2 := plot([seq(t0 + n·(T - t0)/N, n = 1..N)], [seq(y[n], n = 1..N)], color
      = red, legend = Prédateurs);

                          G1 := PLOT(...)
                          G2 := PLOT(...)                          (1.1.1)
> display(G1, G2);
```



4. Recherche de zéros

4.2 La dichotomie

```
> dicho := proc( f, aa, bb, N )
  local a, b, c, fa, fb, fc, n;
  global Xa, Xb :
  a := evalf (aa) :
  b := evalf (bb) :
  c := (a + b) / 2 :
  Xa := array (1 ..N ) :
  Xb := array (1 ..N ) :
  Xa[1] := a;
  Xb[1] := b :
  fa := evalf ( f (a)) :
  fb := evalf ( f (b)) :
```

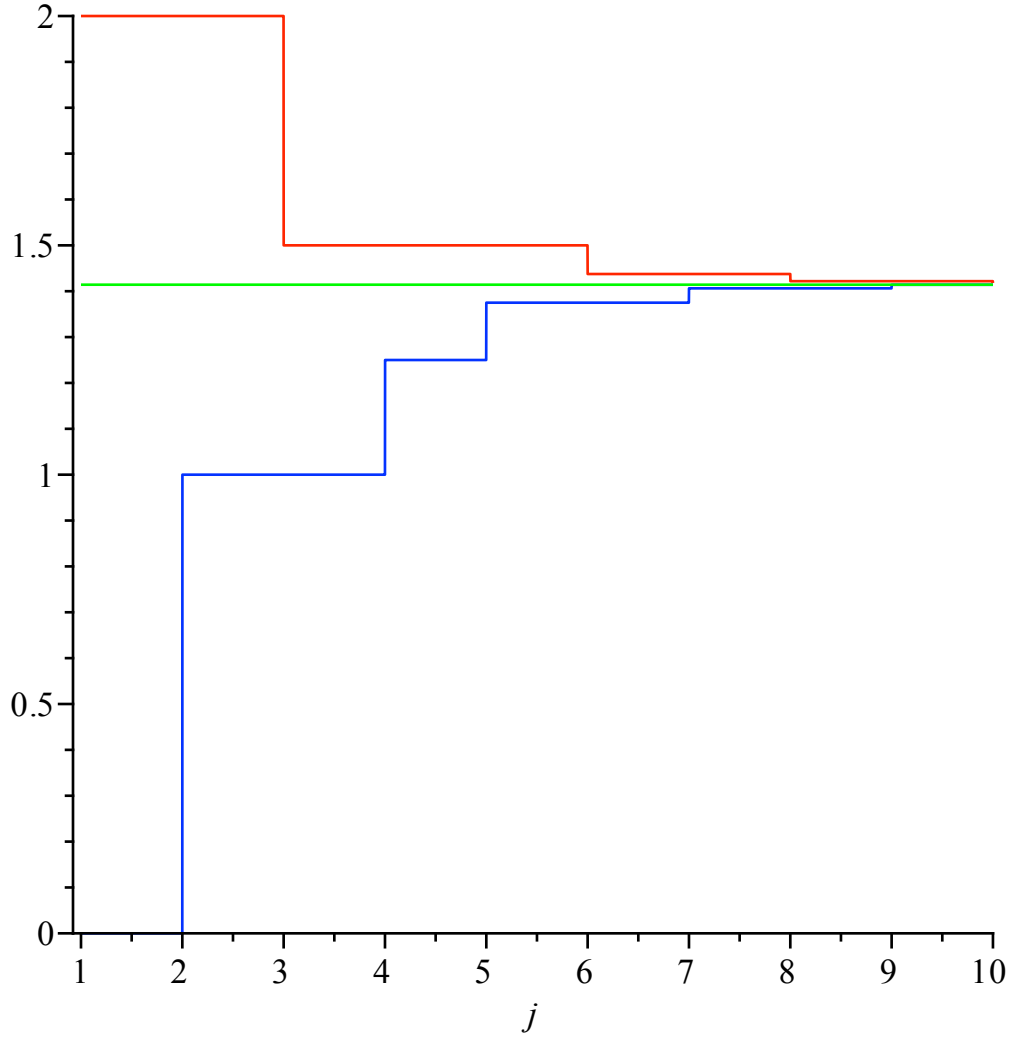
```
    for n from 2 to N do
    fc := evalf( f(c) ) :
    if ( fa·fc < 0 ) then
    Xa[n] := Xa[n − 1] : Xb[n] := c :
    fb := evalf( f(c) ) :
    elif( fc·fb < 0 ) then
    Xa[n] := c : Xb[n] := Xb[n − 1] :
    fa := evalf( f(c) ) :
    else return c : fi:
    c := (Xa[n] + Xb[n]) / 2 :
    od:
    [ [ seq( Xa[n], n = 1 ..N ) ], [ seq( Xb[n], n = 1 ..N ) ] ];
    end proc:
> f := x→4·x + 6 : N := 10 :
    dicho( f, -2, 0, N );
```

$$-1.500000000 \qquad \textbf{(2.1.1)}$$

```
> g := x→x^2 − 2 :
    dicho( g, 0, 2, N );
```

$$[[0., 1.000000000, 1.000000000, 1.250000000, 1.375000000, 1.375000000, 1.406250000, 1.406250000, 1.414062500, 1.414062500], [2., 2., 1.500000000, 1.500000000, 1.500000000, 1.437500000, 1.437500000, 1.421875000, 1.421875000, 1.417968750]] \qquad \textbf{(2.1.2)}$$

```
> plot( [ Xa[ j ], Xb[ j ], sqrt(2) ], j = 1 ..N, color = [ blue, red, green ] );
```



> *evalf* (sqrt (2));

1.414213562 **(2.1.3)**

>

4.3.1 La méthode de la corde

> *corde* := **proc**(*f*, *aa*, *bb*, *N*)
local *a*, *b*, *q*, *k* :
global *X* :
a := *evalf* (*aa*) : *b* := *evalf* (*bb*) :
X := *array* (1 ..*N*) :
$X[1] := \frac{(a+b)}{2}$:
$q := \frac{(evalf(f(b)) - evalf(f(a)))}{b-a}$:
for *k* **from** 1 **to** *N* − 1 **do**

$$X[k+1] := X[k] - \frac{evalf(f(X[k]))}{q} :$$

od:

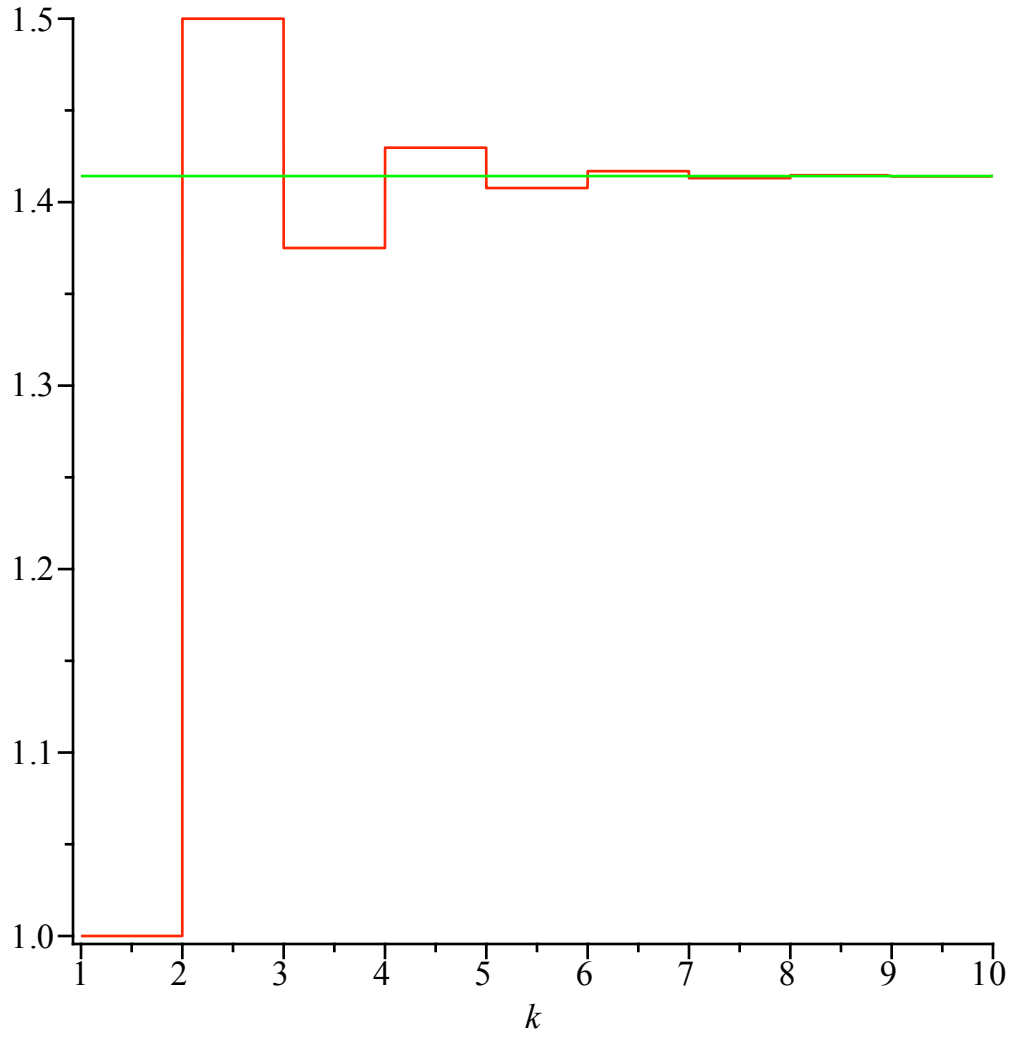
$[seq(X[k], k = 1..N)]$;

end proc:

```
> g := x→x^2 - 2 : N := 10 :
  corde(g, 0, 2, N);
```

[1.000000000, 1.500000000, 1.375000000, 1.429687500, 1.407684326, 1.416896745, 1.413098552, 1.414674793, 1.414022408, 1.414292723] **(2.2.1)**

```
> plot([X[k], sqrt(2)], k = 1..N, color = [red, green]);
```




```
>
>
```

4.3.2 La méthode de la sécante

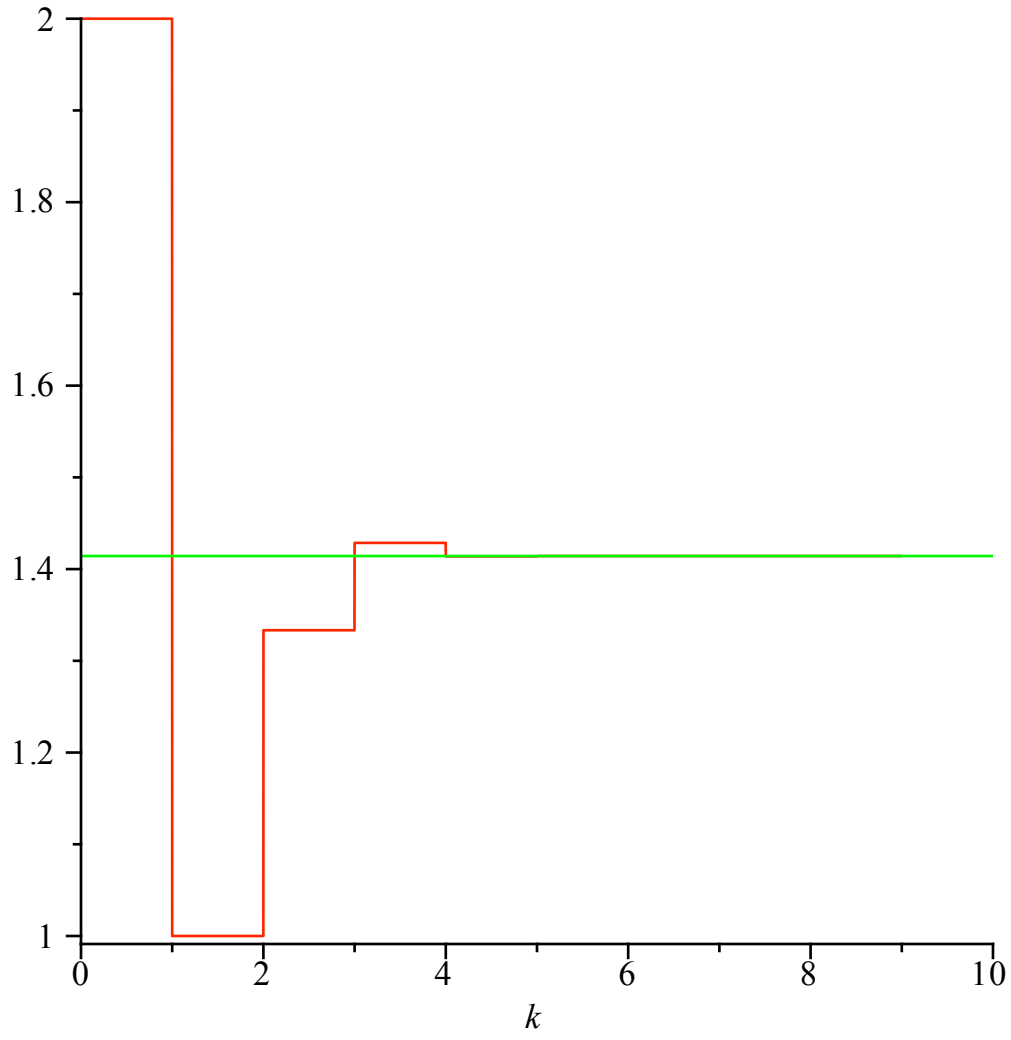
```
> secante := proc(f, aa, bb, N)
  local a, b, q, k :
```

```
global X:
a := evalf(aa): b := evalf(bb):
X := array(-1..N):
X[-1] := a:
X[0] := b:
for k from 0 to N - 1 do
q := (evalf(f(X[k])) - evalf(f(X[k-1]))) / (X[k] - X[k-1]):
X[k+1] := X[k] - evalf(f(X[k])) / q:
od:
[seq(X[k], k=0..N)];
end proc:
```

```
> g := x→x^2 - 2: N := 10:
  secante(g, 0, 2, N);
```

[2., 1.000000000, 1.333333333, 1.428571429, 1.413793103, 1.414211439, 1.414213563, 1.414213562, 1.414213562, *Float*(*undefined*), *Float*(*undefined*)] **(2.3.1)**

```
> plot([X[k], sqrt(2)], k=0..N, color=[red, green]);
```



>

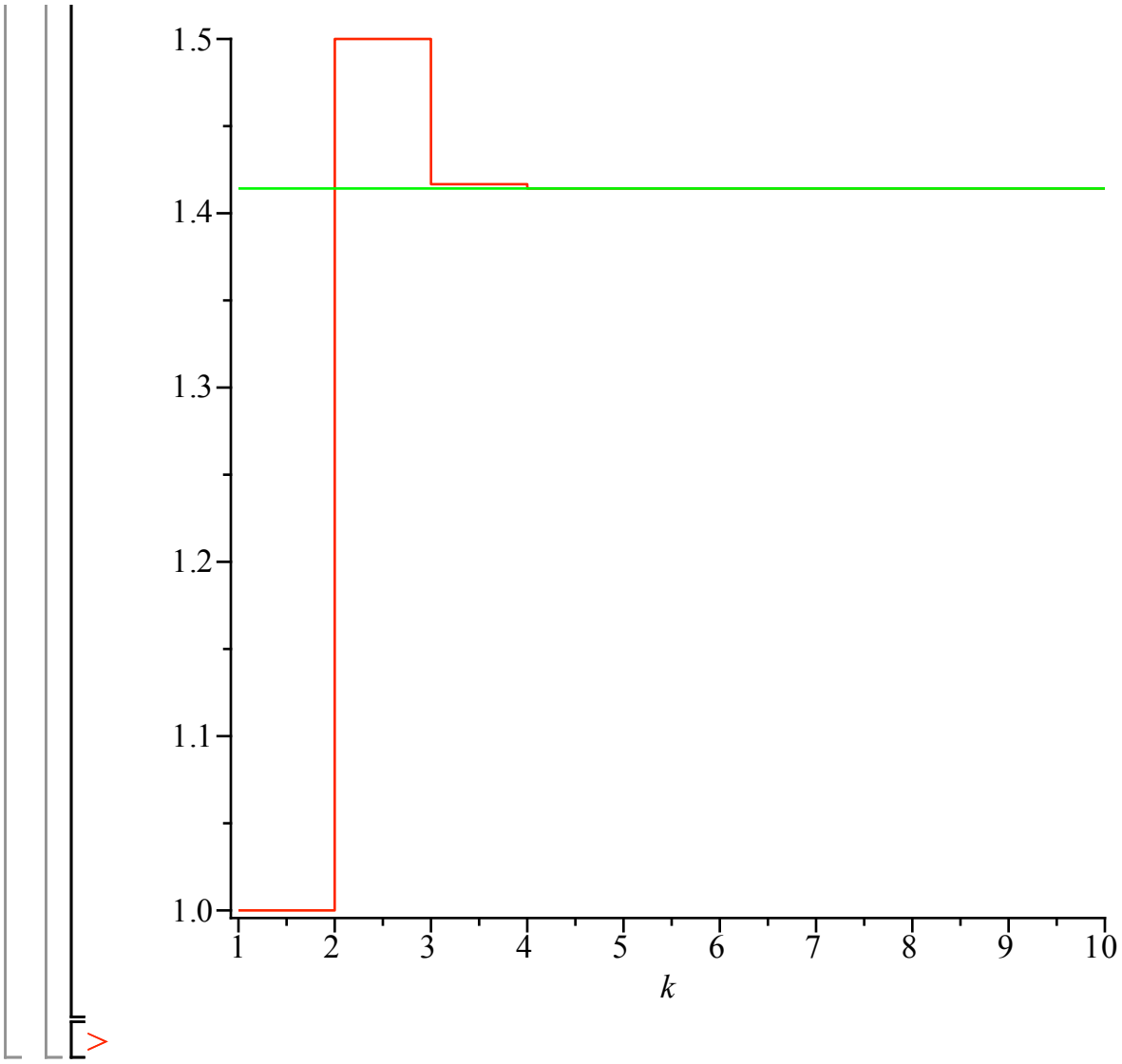
4.3.3 La méthode de Newton

```
> newton := proc( f, df, aa, bb, N )
  local a, b, q, k :
  global X :
  a := evalf (aa) : b := evalf (bb) :
  X := array (1 ..N ) :
  X[1] := (a + b)/2 :
  for k from 1 to N − 1 do
  q := evalf (df (X[k]) ) :
  X[k + 1] := X[k] − evalf ( f (X[k]) )/q :
  od:
  [ seq(X[k], k = 1 ..N ) ];
  end proc:
```

```
> g := x→x^2 − 2 : dg := x→2·x : N := 10 :
  newton( g, dg, 0, 2, N );
```

[1.000000000, 1.500000000, 1.416666667, 1.414215686, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562] **(2.4.1)**

```
> plot( [ X[k], sqrt(2) ], k = 1 ..N, color = [red, green] );
```


1.5
1.4
1.3
1.2
1.1
1.0
1
2
3
4
5
6
7
8
9
10
k