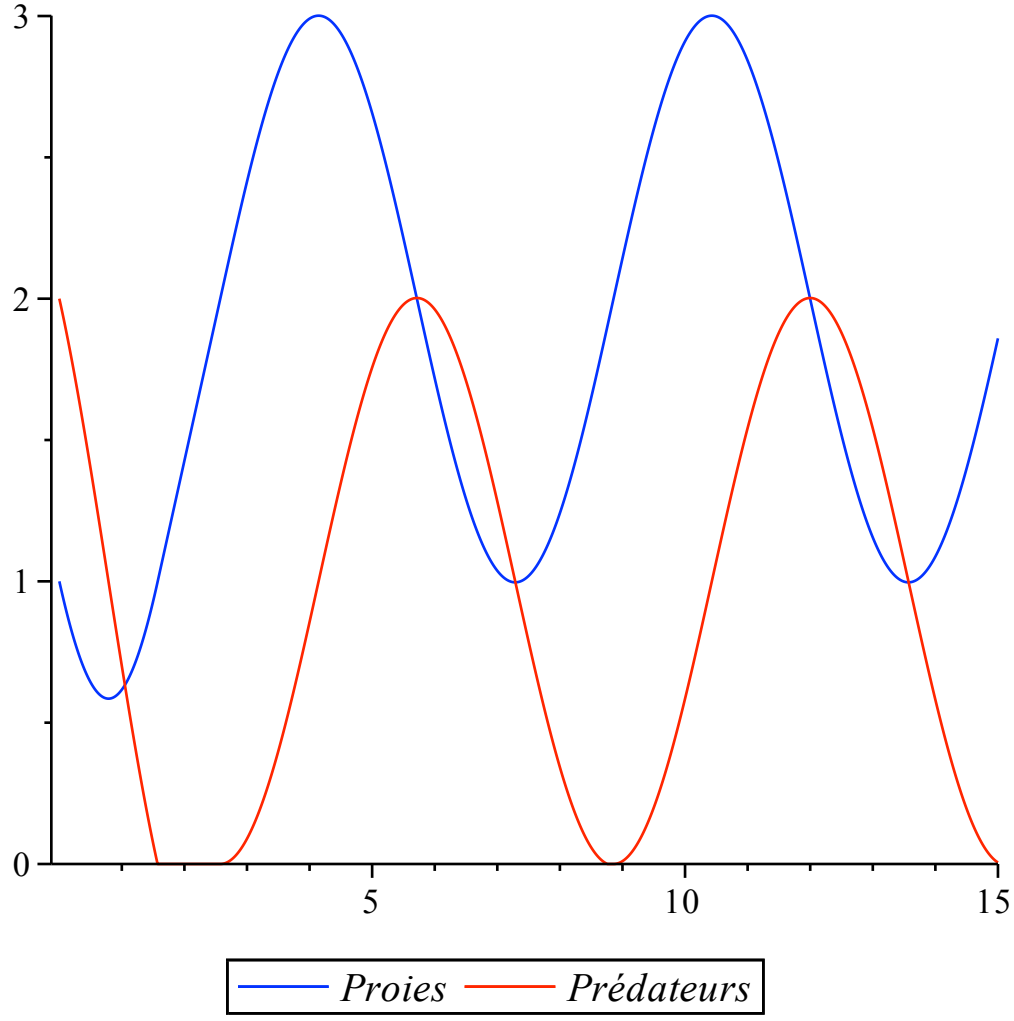
5.7 Exercices

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

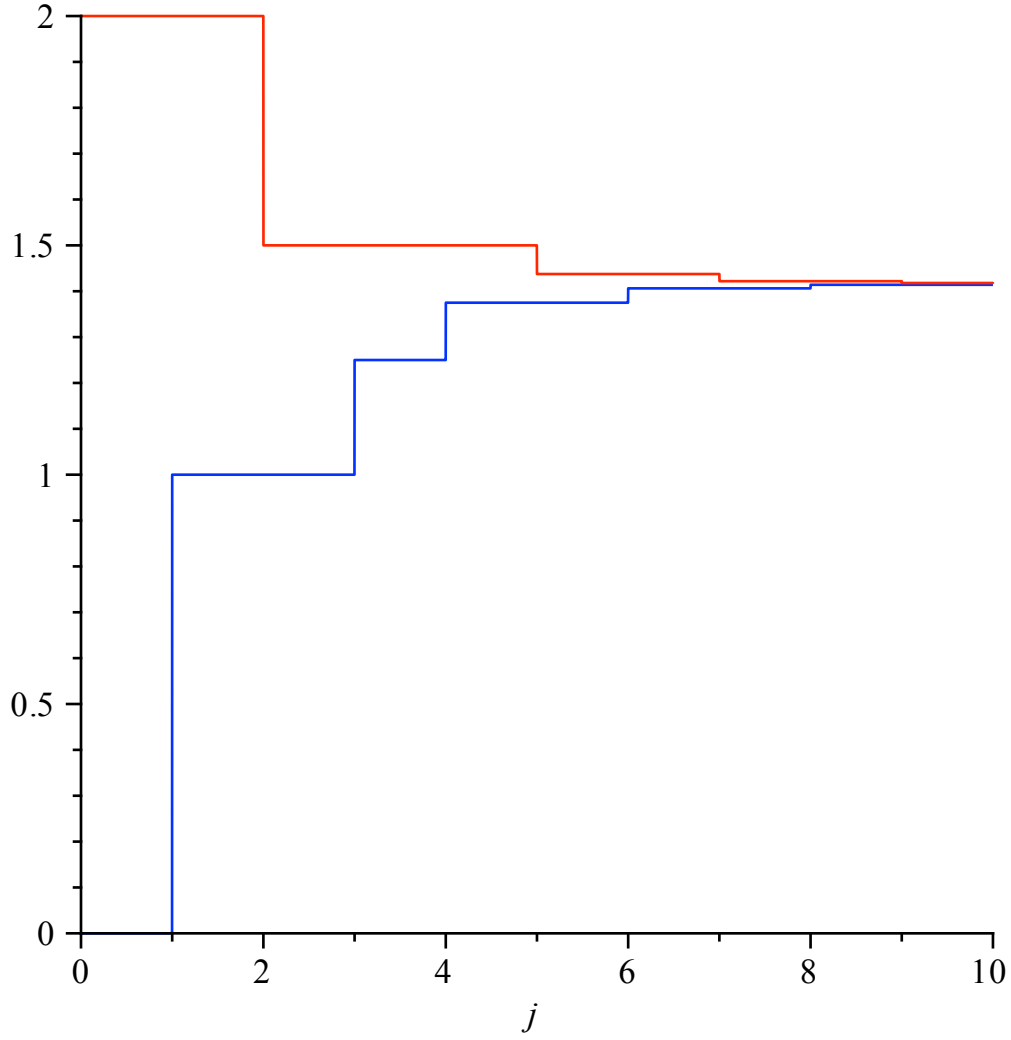
4.2 Dichotomie

```
> dicho := proc( f, aa, bb, N )
  local a, b, c, fa, fb, fc, i :
  global Xa, Xb :
  a := evalf (aa) :
  b := evalf (bb) :
  Xa := array (0..N ) :
  Xb := array (0..N ) :
  Xa[0] := a :
  Xb[0] := b :
  fa := evalf ( f (a) ) :
  fb := evalf ( f (b) ) :
  c := (a + b) / 2 :
```

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

[[0., 1.000000000, 1.000000000, 1.250000000, 1.375000000, 1.375000000, 1.406250000, 1.406250000, 1.414062500, 1.414062500, 1.414062500], [2., 2., 1.500000000, 1.500000000, 1.500000000, 1.437500000, 1.437500000, 1.421875000, 1.421875000, 1.417968750, 1.416015625]] **(2.1.1)**

```
> plot([Xa[j], Xb[j]], j = 0..N, color = [blue, red]);
```



>

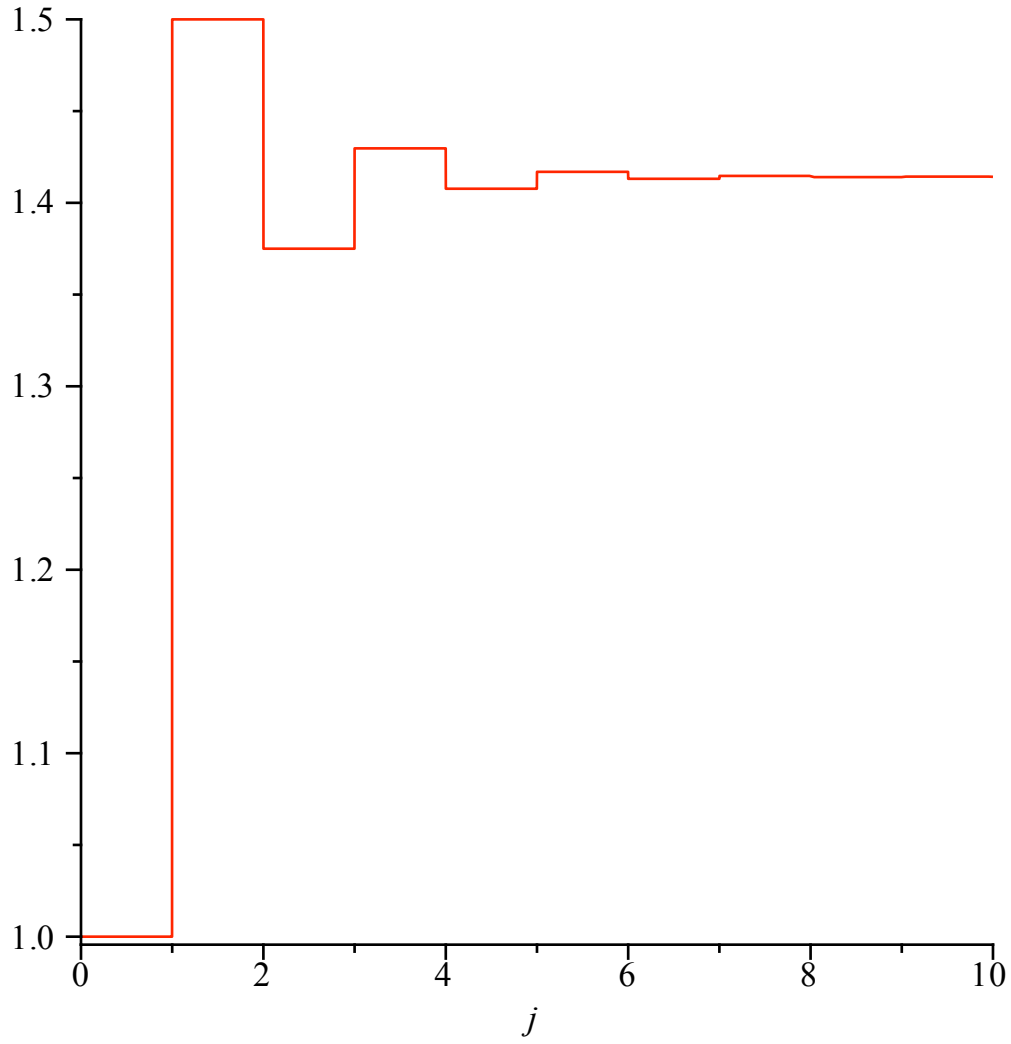
4.3.1 La méthode de la corde

```
> corde := proc( f, aa, bb, x0, N )
  local a, b, fa, fb, k :
   global X :
   a := evalf ( aa ) :
   b := evalf ( bb ) :
   fa := evalf ( f ( a ) ) :
   fb := evalf ( f ( b ) ) :
   X := array ( 0 ..N ) :
   X[ 0 ] := x0 :
   for k from 0 to N − 1 do
  X[ k + 1 ] := X[ k ] − (b − a)/(fb − fa) · evalf ( f ( X[ k ] ) ) :
   od:
   [ seq ( X[ k ], k = 0 ..N ) ] :
   end proc:
```

```
> f := x→x^2 − 2 : N := 10 :
  corde( f, 0, 2, 1, N );
```

[1, 1.500000000, 1.375000000, 1.429687500, 1.407684326, 1.416896745, 1.413098552, 1.414674793, 1.414022408, 1.414292723, 1.414180770] **(2.2.1)**

```
> plot( X[ j ], j = 0 ..N );
```




```
>
```

4.3.2 La méthode de la sécante

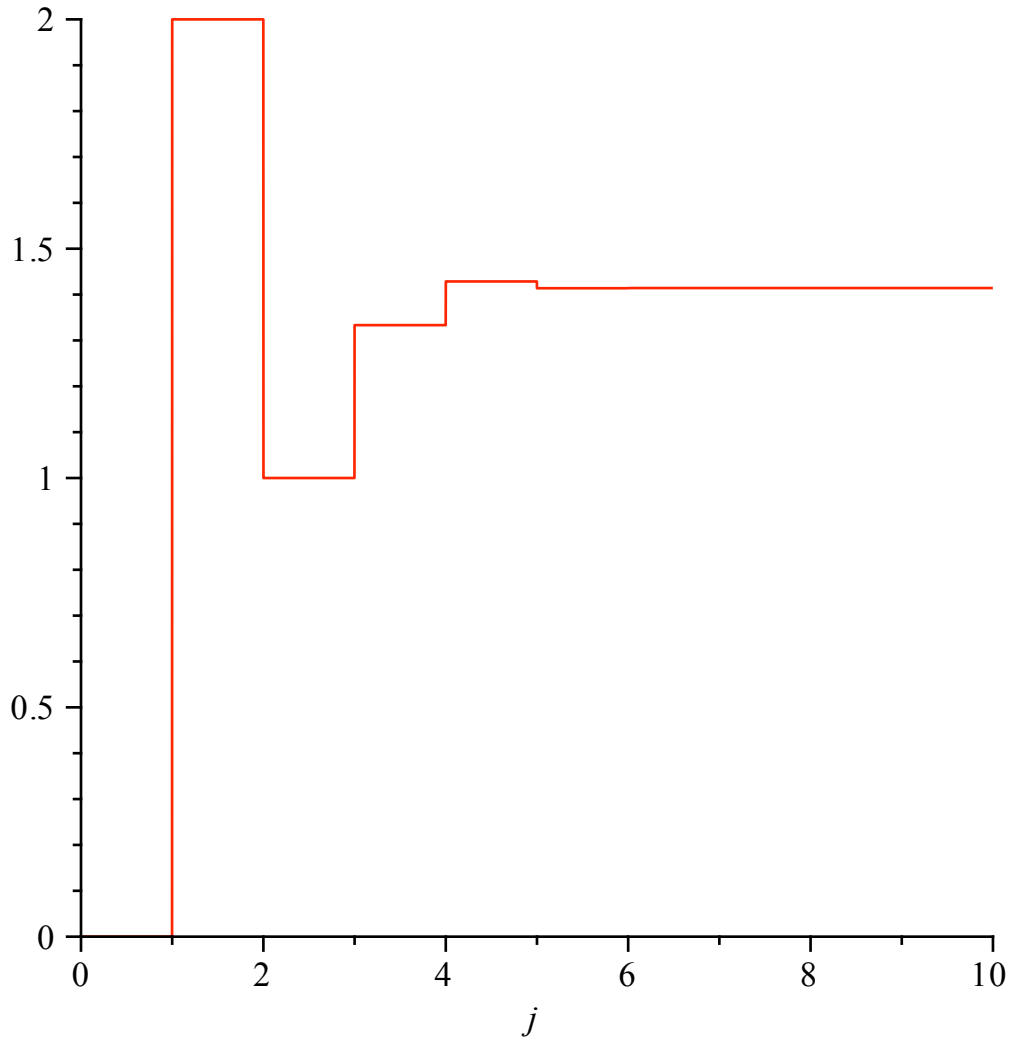
```
> secante := proc( f, aa, bb, N )
  local a, b, f1, f2, k, q :
   global X :
   a := evalf (aa) :
   b := evalf (bb) :
   f1 := evalf ( f (a) ) :
   f2 := evalf ( f (b) ) :
   X := array (0 ..N ) :
```

```
X[0] := a :
X[1] := b :
for k from 1 to N − 1 do
q := (f2 − f1) / (X[k] − X[k − 1]) :
X[k + 1] := X[k] − (1/q) · evalf(f(X[k])) :
f1 := f2 :
f2 := evalf(f(X[k + 1])) :
od:
[seq(X[k], k = 0..N)] :
end proc:
```

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

[0., 2., 1.000000000, 1.333333333, 1.428571429, 1.413793103, 1.414211439, 1.414213563, 1.414213562, 1.414213562, *Float*(*undefined*)] **(2.3.1)**

```
> plot(X[j], j = 0..N);
```




```
>
```

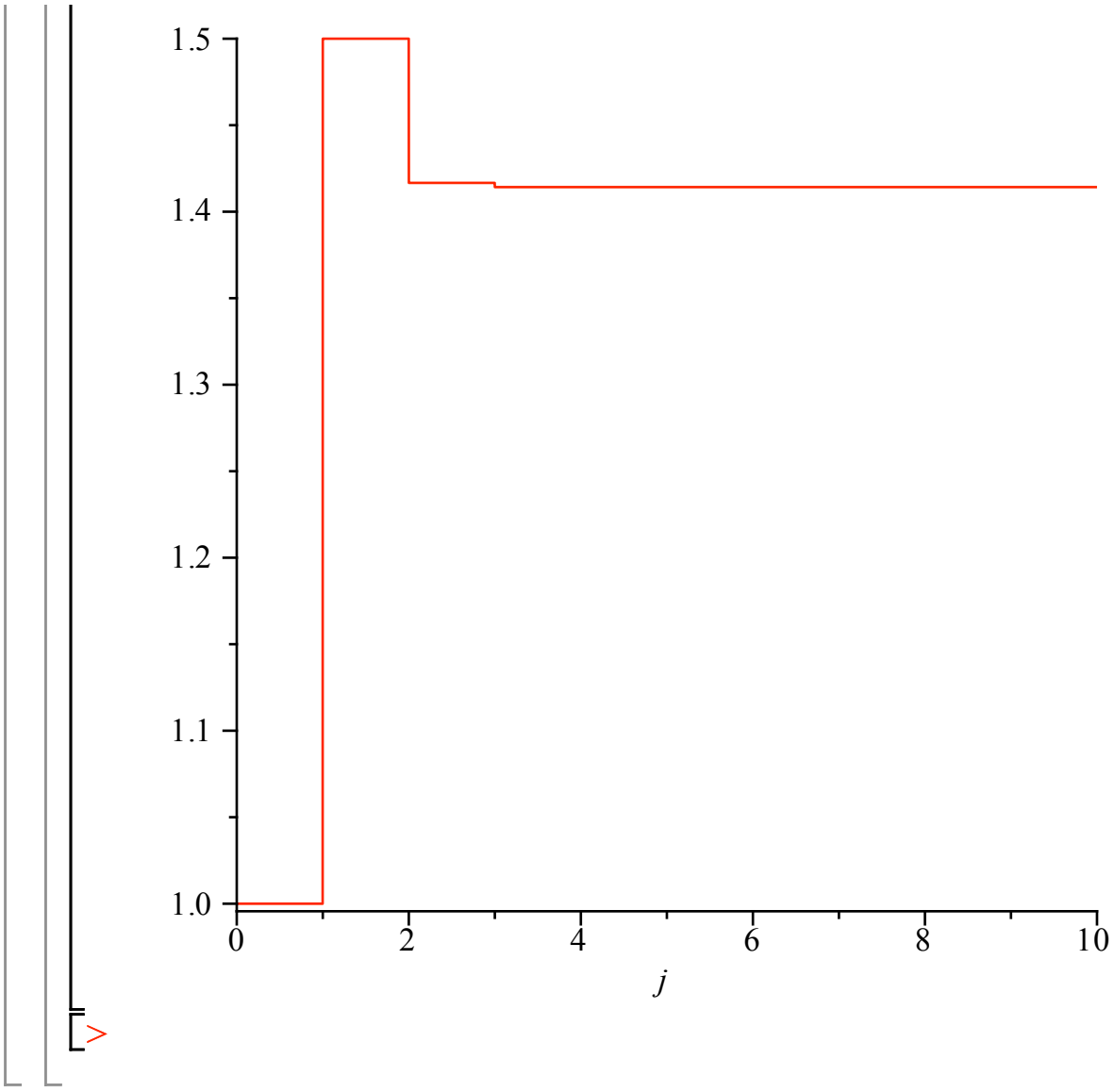
4.3.4 La méthode de Newton

```
> Newton := proc( f, df, xx0, N )
  local f1, x0, k, q :
   global X :
   x0 := evalf ( xx0 ) :
   f1 := evalf ( f ( x0 ) ) :
   X := array ( 0 ..N ) :
   X [ 0 ] := x0 :
   for k from 0 to N − 1 do
   q := evalf ( df ( X [ k ] ) ) :
  X [ k + 1 ] := X [ k ] − 1/q · evalf ( f ( X [ k ] ) ) :
   od:
   [ seq ( X [ k ], k = 0 ..N ) ] :
   end proc:
```

```
> f := x→x^2 − 2 : df := x→2·x : N := 10 :
   Newton( f, df, 1, N );
```

$$[1., 1.500000000, 1.416666667, 1.414215686, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562] \quad \textbf{(2.4.1)}$$

```
> plot ( X [ j ], j = 0 ..N );
```


1.5
1.4
1.3
1.2
1.1
1.0
0
2
4
6
8
10
j