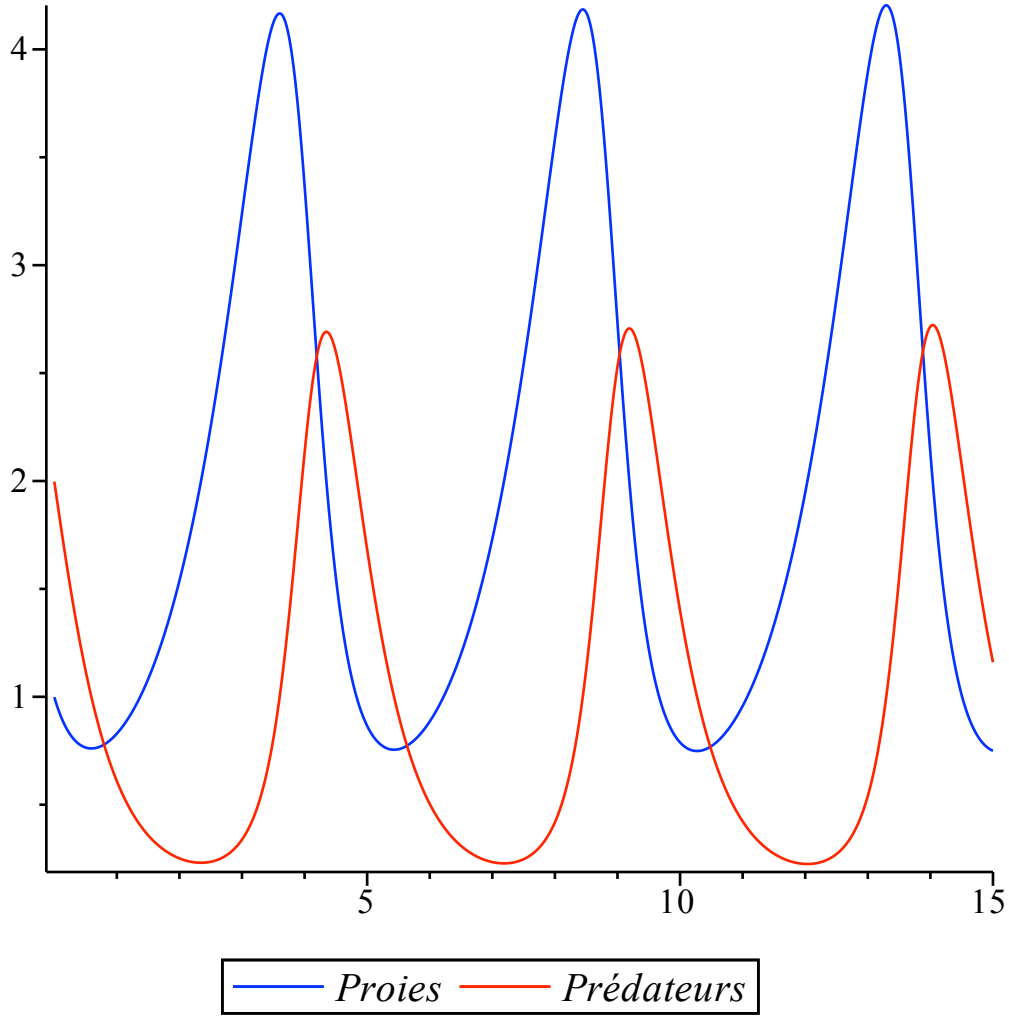
5.5 Le cas de systèmes d'équations différentielles

Exercice 33

```
> restart; digits := 100 :
> lotka := proc(x0, y0, t0, T, N)
  local h, alpha, beta, delta, gam, n :
  global x, y :
  h := (T - t0)/N :
  alpha := 1 : beta := 1 : delta := 1 : gam := 2 :
  x := array(0..N) : y := array(0..N) :
  x[0] := x0 : y[0] := y0 :
  for n from 0 to N - 1 do
  x[n + 1] := evalf(x[n] + h*x[n]*(alpha - beta*y[n])) :
  y[n + 1] := evalf(y[n] - h*y[n]*(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 : T := 15 : t0 := 0 :
> lotka(1, 2, t0, T, N) :
> with(plots) :
> h := (15 - 0)/N :
> 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);
```



>

4.2 La méthode de la bisection

```
> dicho := proc( f, aa, bb, N )
  local a, b, c, fa, fb, fc, n :
  global Xa, Xb :
  a := evalf (aa) : b := evalf (bb) :
  Xa := array (1 ..N ) : Xb := array (1 ..N ) :
  Xa[ 1 ] := a :
  Xb[ 1 ] := b :
  c := (a + b) / 2 :
  fa := evalf ( f (a) ) : fb := evalf ( f (b) ) :
  for n from 1 to N − 1 do
 fc := evalf ( f (c) ) :
  if ( fa·fc < 0 ) then
  Xa[ n + 1 ] := Xa[ n ] : Xb[ n + 1 ] := c :
```

```
    fb := fc :
    elif ( fc·fb < 0) then
    Xa[n + 1] := c : Xb[n + 1] := Xb[n] :
    fa := fc :
    else return c :
    fi:
    c := (Xa[n + 1] + Xb[n + 1]) / 2 :
    od:
    [ [ seq( Xa[i], i = 1 ..N ) ], [ seq( Xb[i], i = 1 ..N ) ] ] :
    end proc:
```

```
> f := x→4 x + 6 :
  dicho( f, -4, 0, 10);
```

$$-1.500000000 \qquad (2.1)$$

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

$$[[0., 1.000000000, 1.000000000, 1.250000000, 1.375000000, 1.375000000, 1.406250000, 1.406250000, 1.414062500, 1.414062500], [2., 2., 1.500000000, 1.500000000, 1.500000000, 1.437500000, 1.437500000, 1.421875000, 1.421875000, 1.417968750]] \qquad (2.2)$$

```
> G1 := plot( Xa[n], n = 1 ..N, color = blue, legend = "an");
```

$$G1 := PLOT(\ldots) \qquad (2.3)$$

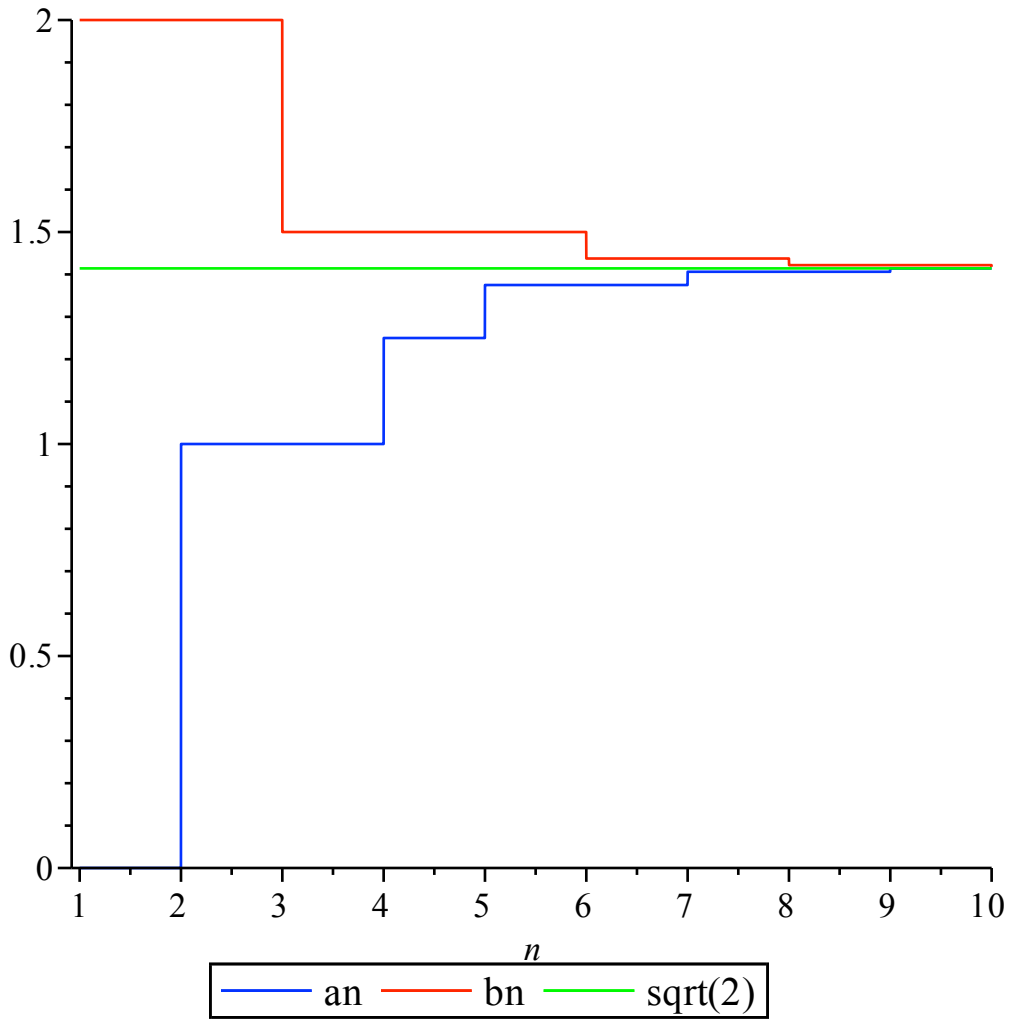
```
> G2 := plot( Xb[n], n = 1 ..N, color = red, legend = "bn");
```

$$G2 := PLOT(\ldots) \qquad (2.4)$$

```
> G3 := plot(sqrt(2), n = 1 ..N, color = green, legend = "sqrt(2)");
```

$$G3 := PLOT(\ldots) \qquad (2.5)$$

```
> display(G1, G2, G3);
```



>

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] := (a + b) / 2 :
  q := (f (b) − f (a)) / (b − a) :
  for k from 1 to N − 1 do
  X[k + 1] := X[k] − evalf ( f (X[k]) / q ) :
```

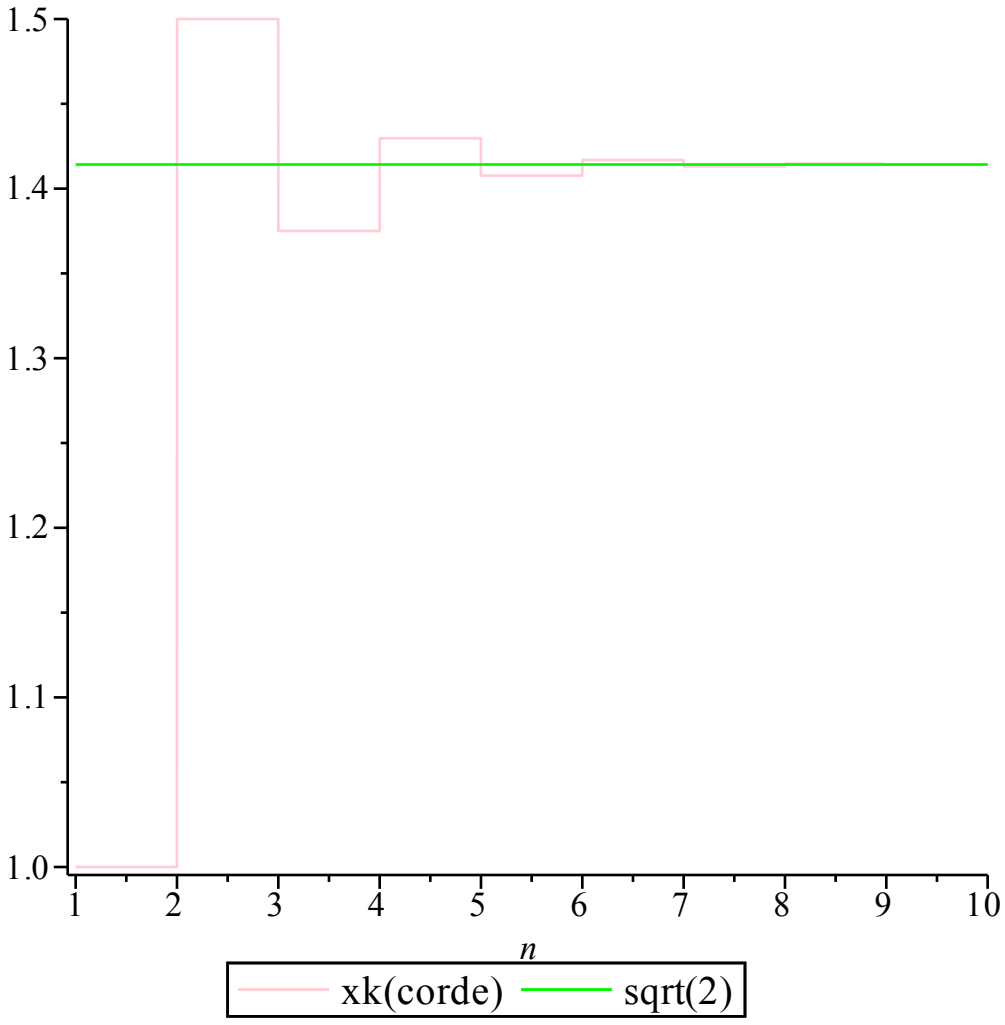
```
od:
[seq(X[k], k=1..N)];
end proc:
>
> f := x→x^2 − 2 : N := 10 :
corde(f, 0, 2, N);
```

[1.000000000, 1.500000000, 1.375000000, 1.429687500, 1.407684326, 1.416896745, 1.413098552, 1.414674793, 1.414022408, 1.414292723] **(3.1)**

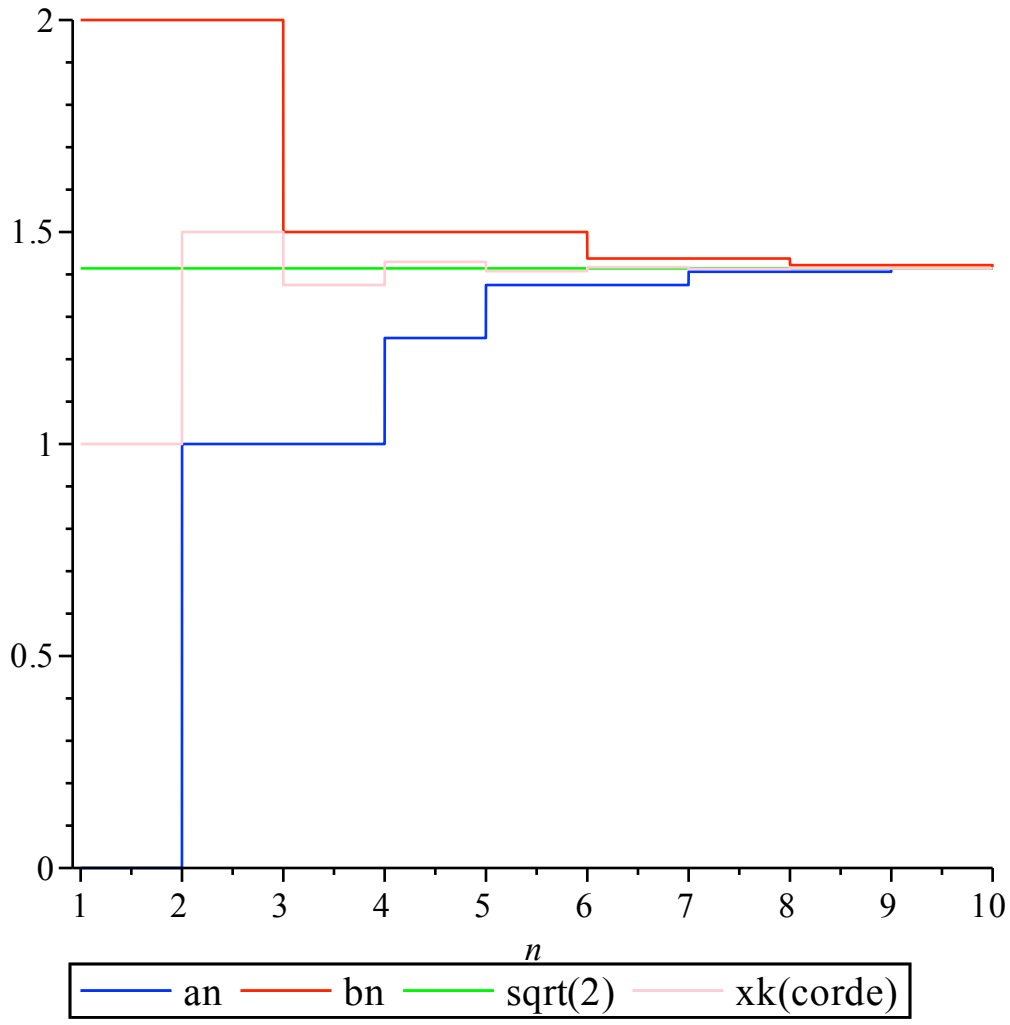
```
> G4 := plot(X[n], n=1..N, color=pink, legend="xk(corde)");
```

$G4 := PLOT(\ldots)$ **(3.2)**

```
> display(G4, G3);
```




```
> #Comparaison des méthodes de la corde et de la dichotomie
> display(G1, G2, G3, G4);
```




```
> #La méthode de la corde est plus précise ici.
```

4.3.2 La méthode de la sécante

```
> secante := proc( f, aa, bb, N )
  local a, b, qk, k;
  global X:
  a := evalf (aa) :
  b := evalf (bb) :
  X := array (0 ..N ) :
  X[0] := a :
  X[1] := b :
  for k from 1 to N − 1 do
  qk := (f (X[k]) − f (X[k − 1])) / (X[k] − X[k − 1]) :
  X[k + 1] := X[k] − evalf ( f (X[k]) / qk ) :
  od:
```

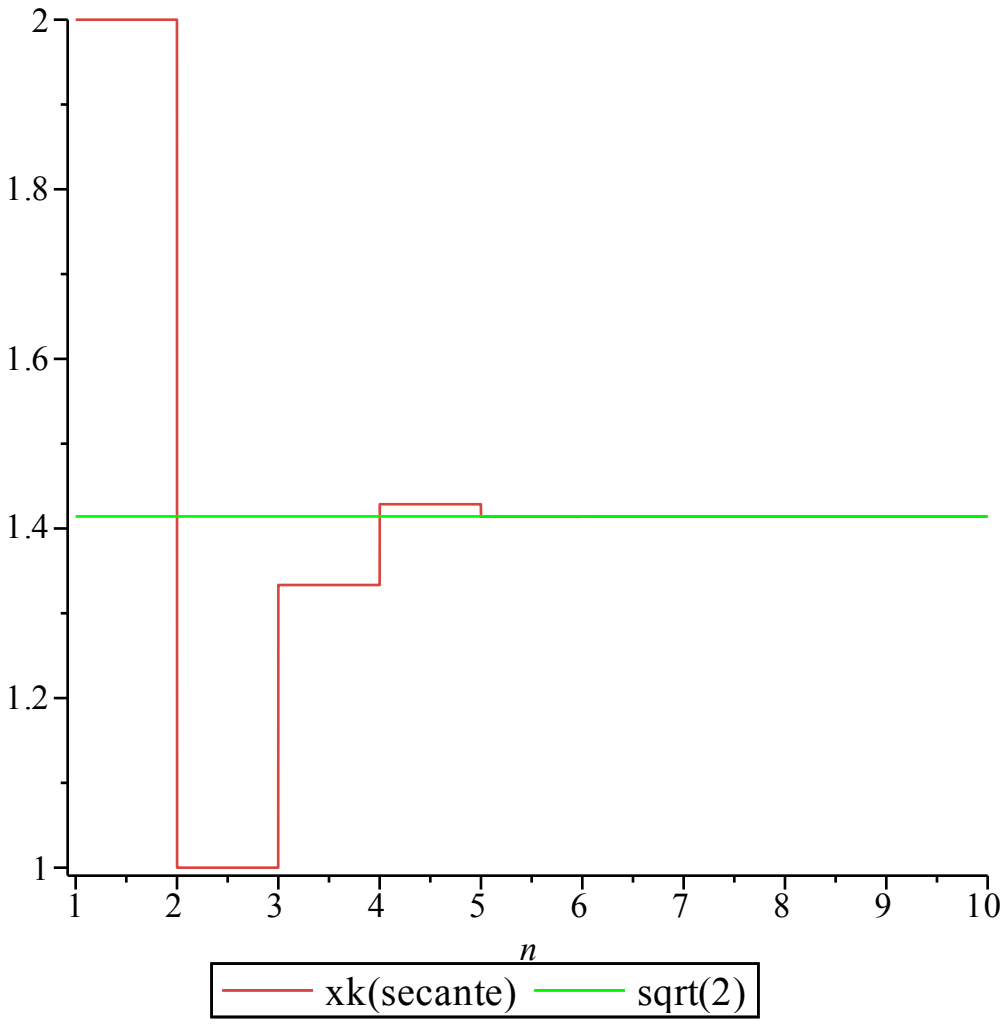
```
    [ seq( X[k], k = 1 ..N ) ];
    end proc:
>
> f := x→x^2 − 2 : N := 10 :
    secante( f, 0, 2, N );
```

[2., 1.000000000, 1.333333333, 1.428571429, 1.413793103, 1.414211439, 1.414213563, 1.414213562, 1.414213562, *Float*(*undefined*)] **(4.1)**

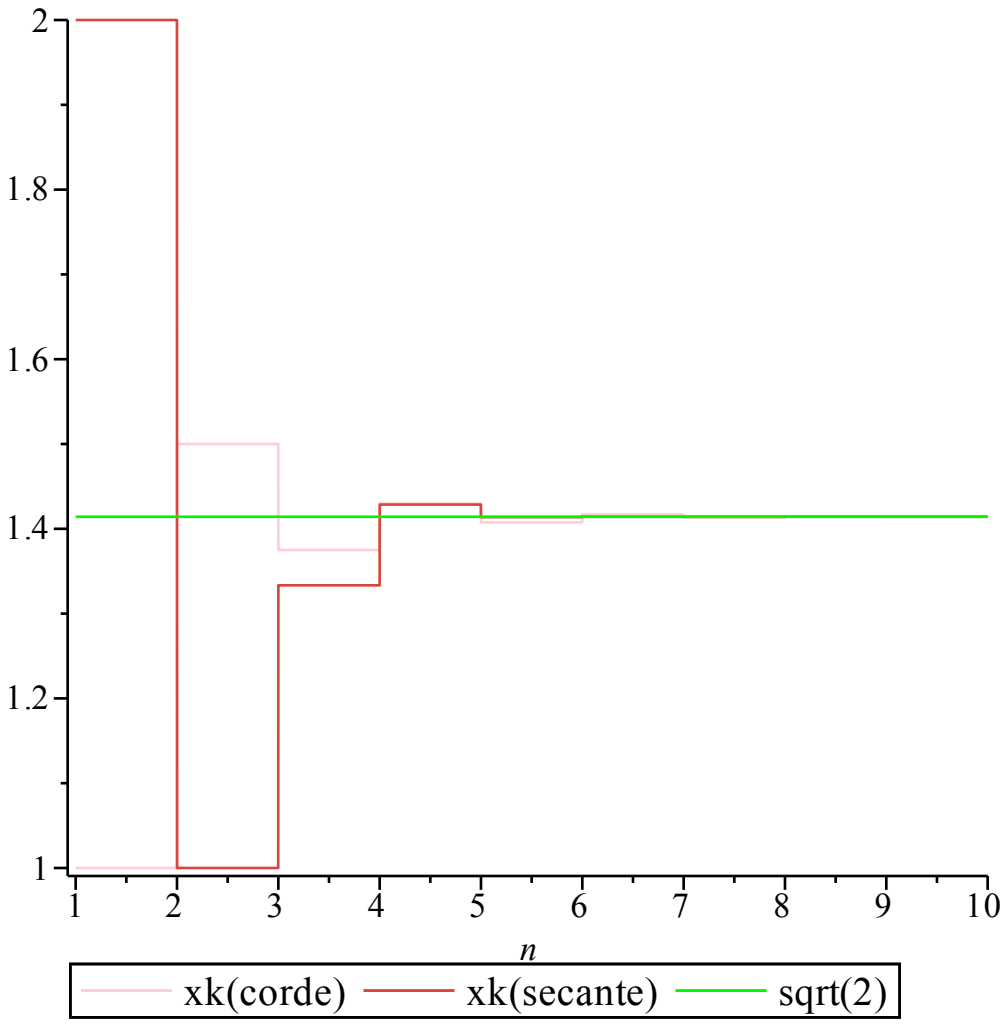
```
> G5 := plot( X[n], n = 1 ..N, color = orange, legend = "xk(secante)" );
```

G5 := *PLOT*(...) **(4.2)**

```
> display( G5, G3 );
```




```
> #Comparaison des méthodes de la corde et de la sécante
> display( G4, G5, G3 );
```



> *#La méthode de la sécante a l'air de tendre plus rapidemment vers la solution exacte (elles sont "confondues" sur le graphe à partir de k=5 alors que c'est k=7 pour la méthode de la corde).*

4.3.3 La méthode de Newton

```
> newton := proc( f, df, aa, bb, N )
  local a, b, qk, 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
  qk := df (X[k]) :
  X[k + 1] := X[k] − evalf ( f (X[k])/qk ) :
  od:
```

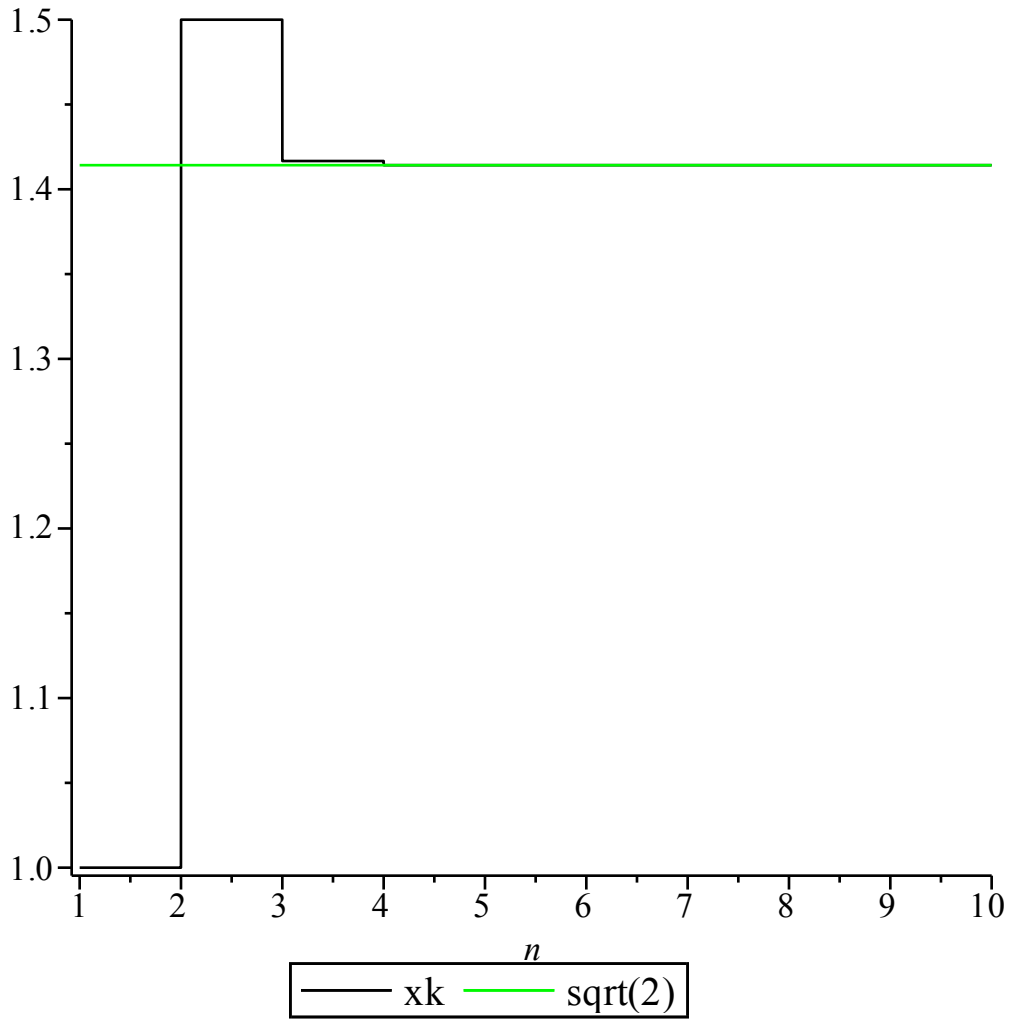
```
    [ seq( X[k], k = 1 ..N ) ];
    end proc:
>
> f := x→x^2 − 2 : df := x→2·x : N := 10 :
    newton( f, df, 0, 2, N );
```

[1.000000000, 1.500000000, 1.416666667, 1.414215686, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562, 1.414213562] **(5.1)**

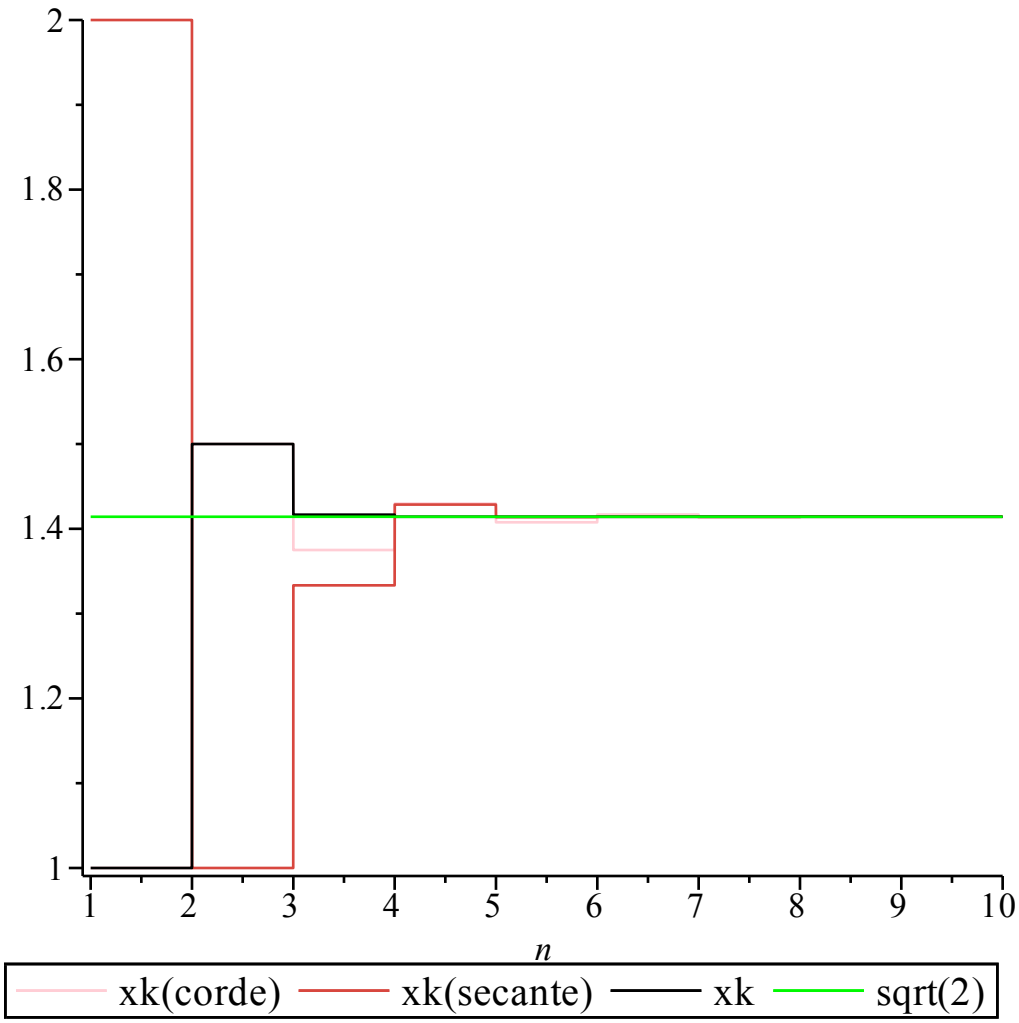
```
> G6 := plot( X[n], n = 1 ..N, color = black, legend = "xk" );
```

G6 := PLOT(...) **(5.2)**

```
> display( G6, G3 );
```




```
> #Comparaison des méthodes de la corde, de la sécante et de Newton
> display( G4, G5, G6, G3 );
```



> *#La méthode de la Newton a l'air de tendre plus rapidemment vers la solution exacte (elles sont* "confondues" *à partir de k = 4 sur le graphe alors que c'est k = 5 pour la méthode de la sécante et k = 7 pour la méthode de la corde).*

>