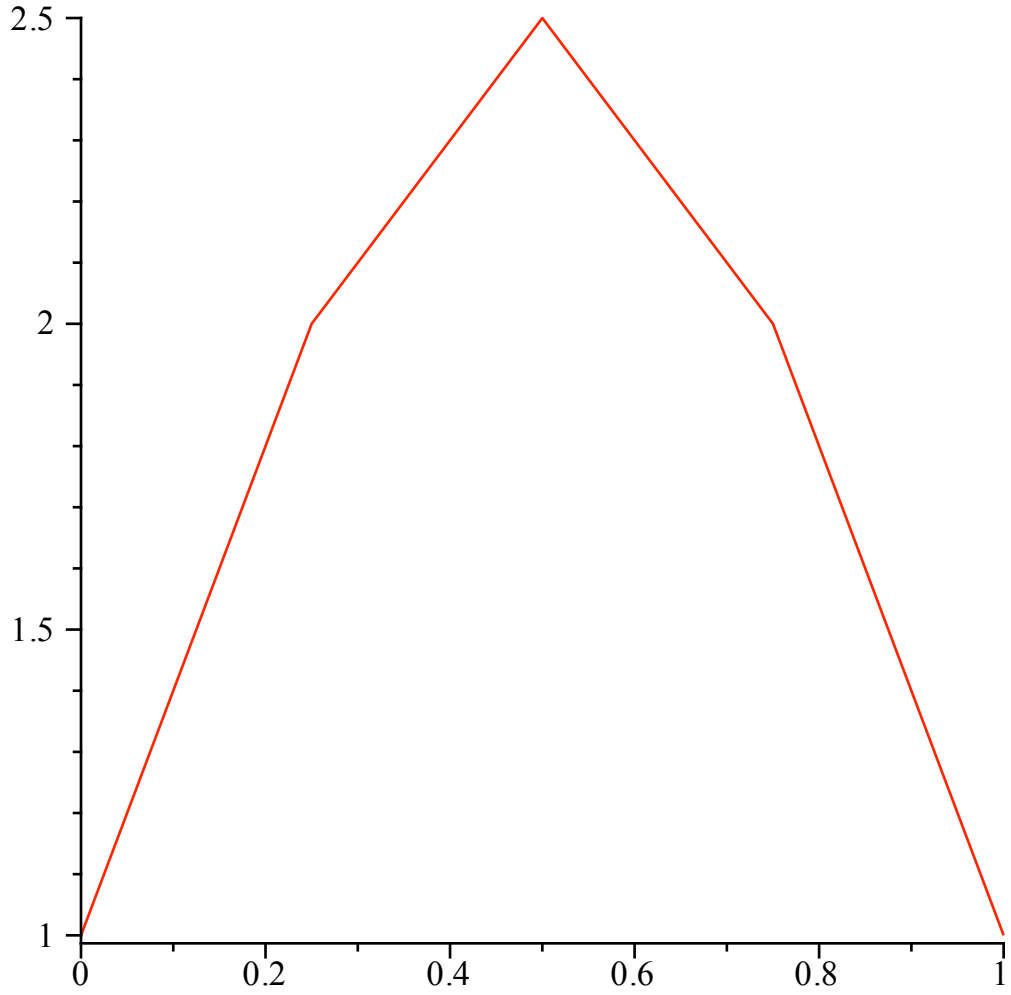
2. Interpolation

2.2.2 Algorithme

```
> restart :
> interpolin := proc( fX, x )
  local N, k, alpha :
  N := nops( fX ) - 1 :
  if ( x = 1 ) then
  return fX [ N + 1 ] :
  fi:
  k := floor ( N·x ) :
  alpha := x·N - k :
  fX [ k + 1 ] + ( fX [ k + 2 ] - fX [ k + 1 ]) ·alpha :
  end proc:
> interpolin( [ 1, 2, 2.5, 2, 1 ], 0.3 );
```

2.10 **(1.1.1)**

```
> plot( [ [ 0, 1 ], [ 0.25, 2 ], [ 0.5, 2.5 ], [ 0.75, 2 ], [ 1, 1 ] ] );
```



2.2.3 Exercices

Exercice 4

> $T := 23;$

$$T := 23 \qquad \textbf{(1.2.1.1)}$$

> $interpolin\left([1.308, 1.141, 1.005, 0.804, 0.727, 0.661], \frac{(T-10)}{40-10} \right);$

$$0.9715000000 \qquad \textbf{(1.2.1.2)}$$

Exercice 5

```
> #Première methode
```

```
> interpolin2 := proc( fX, x, a, b )
  interpolin( fX, (x - a)/(b - a) ) :
  end proc:
```

```
> interpolin2( [1.308, 1.141, 1.005, 0.804, 0.727, 0.661], T, 10, 40);
```

0.9715000000 **(1.2.2.1)**

```
> #Deuxième méthode
```

```
> interpolin2bis := proc( fX, x, a, b )
  local N, h, k :
  N := nops( fX ) - 1 :
  if (x = b) then
  return fX[N + 1] :
  fi:
  h := (b - a)/N :
  k := floor( (x - a)/h ) :
  fX[k + 1] + (fX[k + 2] - fX[k + 1])/h · (x - (a + k·h)) :
  end proc:
```

```
> interpolin2bis( [1.308, 1.141, 1.005, 0.804, 0.727, 0.661], T, 10, 40);
```

0.9715000000 **(1.2.2.2)**

```
>
```

Exercice 7

```
> interpolin2( [1.308, 1.141, 1.005, 0.804, 0.727, 0.661], T, 10, 40);
```

0.9715000000 **(1.2.3.1)**

```
>
```

Exercice 8

```
> 362.78 + (512.35 - 362.78)/(20 - 15) · (16 - 15);
```

392.6940000 **(1.2.4.1)**

```
> interpolin2( [0, 227.04, 362.78, 512.35, 602.97, 901.67], 16, 0, 30);
```

462.4933333 **(1.2.4.2)**

```
> interpolin2( [362.78, 512.35], 16, 15, 20);
```

392.6940000 **(1.2.4.3)**

```
>
```

Exercice 9

```
> localisation := proc(X, x)
  local N, i :
  N := nops(X) - 1 :
  if (x = X[N + 1]) then
  return N :
  fi:
  i := 1 :
  while (X[i + 1] ≤ x) do
  i := i + 1 :
  od:
  i := i - 1 :
  end proc:
> localisation([0, 1, 15, 15.5, 16, 56, 194], 76);
```

5 **(1.2.5.1)**

```
>
```

Exercice 10

```
> interpolin3 := proc(X, fX, x)
  local N, i :
  N := nops(fX) - 1 :
  i := localisation(X, x);
  if (i = N) then
  return fX[N + 1] :
  fi:
  interpolin2([fX[i + 1], fX[i + 2]], x, X[i + 1], X[i + 2]);
  end proc:
> interpolin3([0, 10, 15, 20, 22.5, 30], [0, 227.04, 362.78, 512.35, 602.97, 901.67],
      16);
```

392.6940000 **(1.2.6.1)**

```
> interpolin3bis := proc(X, fX, x)
  local N, i :
  N := nops(fX) - 1 :
  i := localisation(X, x);
  if (i = N) then
  return fX[N + 1] :
  fi:
```

$$fX[i+1] + \frac{(fX[i+2] - fX[i+1])}{X[i+2] - X[i+1]} \cdot (x - X[i+1]);$$

```
  end proc:
> interpolin3bis([0, 10, 15, 20, 22.5, 30], [0, 227.04, 362.78, 512.35, 602.97, 901.67],
      16);
```

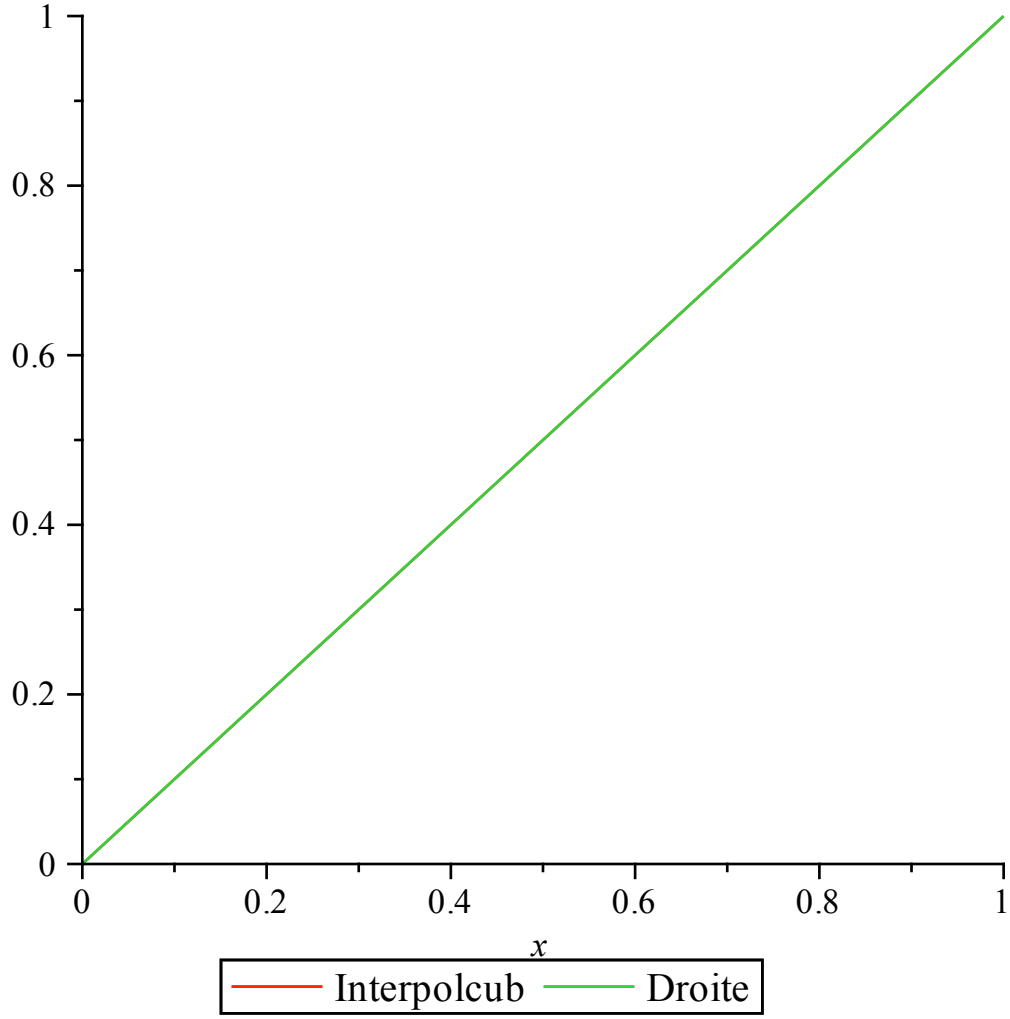
392.6940000 **(1.2.6.2)**

```
>
```

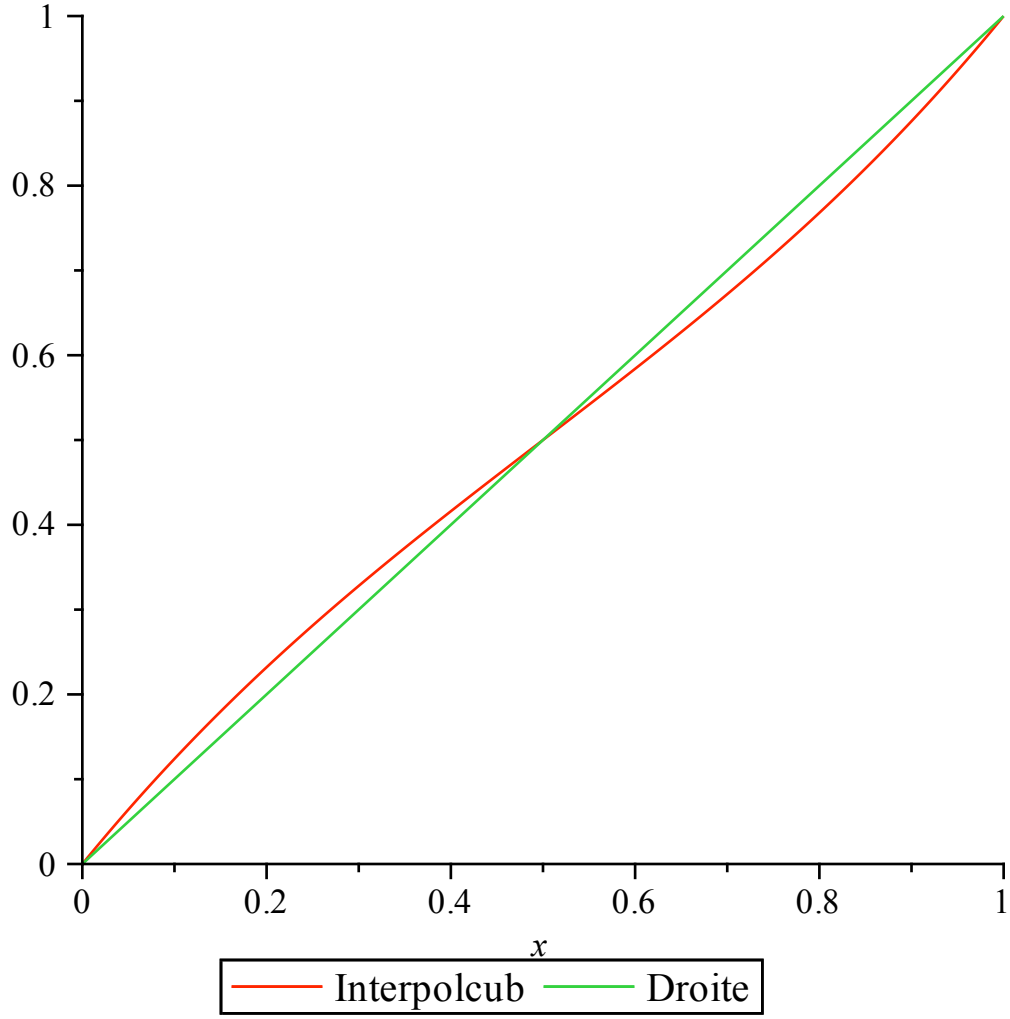
2.3 Interpolation cubique

2.3.3 Algorithme

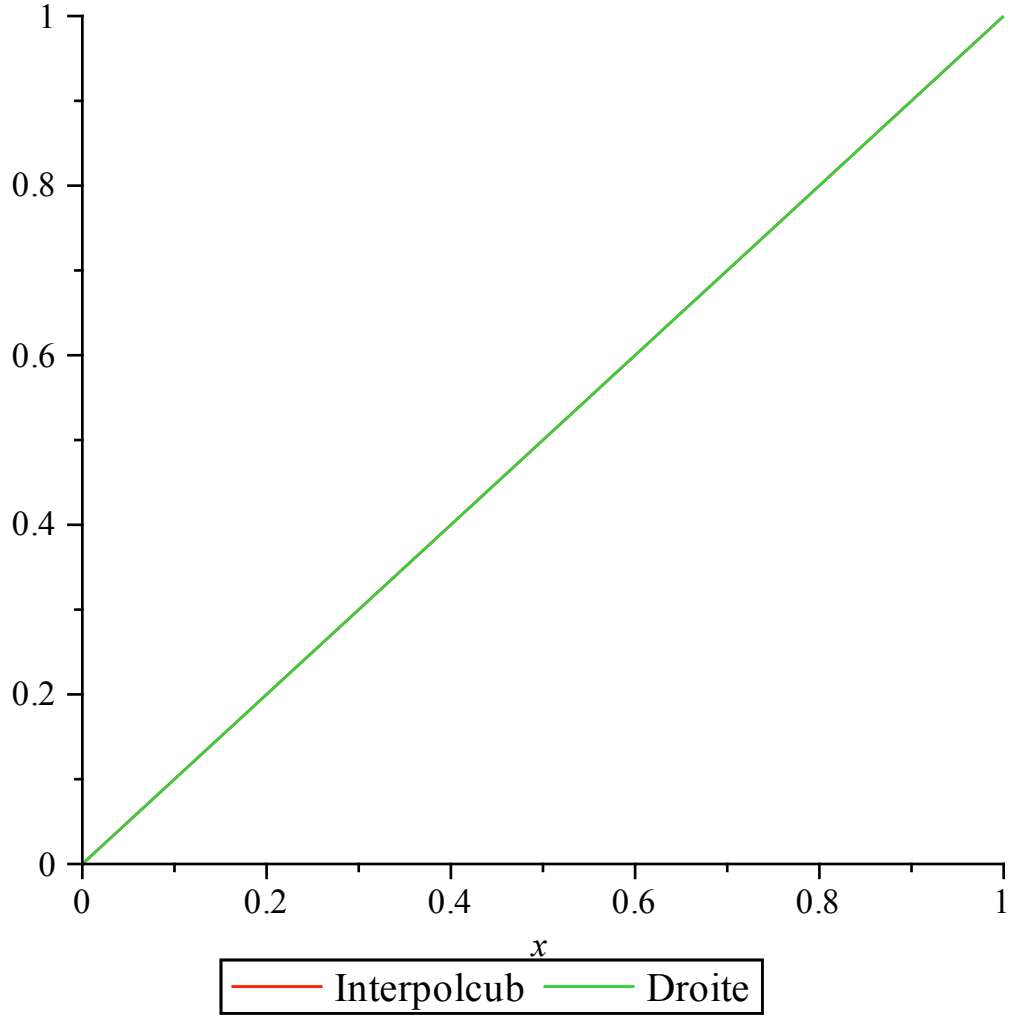
```
> interpolcub := proc( fX, x )
  local N, k, a, s :
  N := nops( fX ) - 3 :
  if ( x = 1 ) then
  return fX[ N + 2 ] :
  fi:
  k := floor( N·x ) :
  a := N·x - k :
  s := - fX[k + 1]·a·(1 - a)·(2 - a) / 6 :
  s := s + fX[k + 2]·(1 + a)·(1 - a)·(2 - a) / 2 :
  s := s + fX[k + 3]·(1 + a)·a·(2 - a) / 2 :
  s := s - fX[k + 4]·(1 + a)·a·(1 - a) / 6 :
  end proc:
> f := interpolcub( [ -1, 0, 1, 2 ], x ) :
  plot( [ f, x ], x = 0 ..1, legend = [ "Interpolcub", "Droite" ] );
```



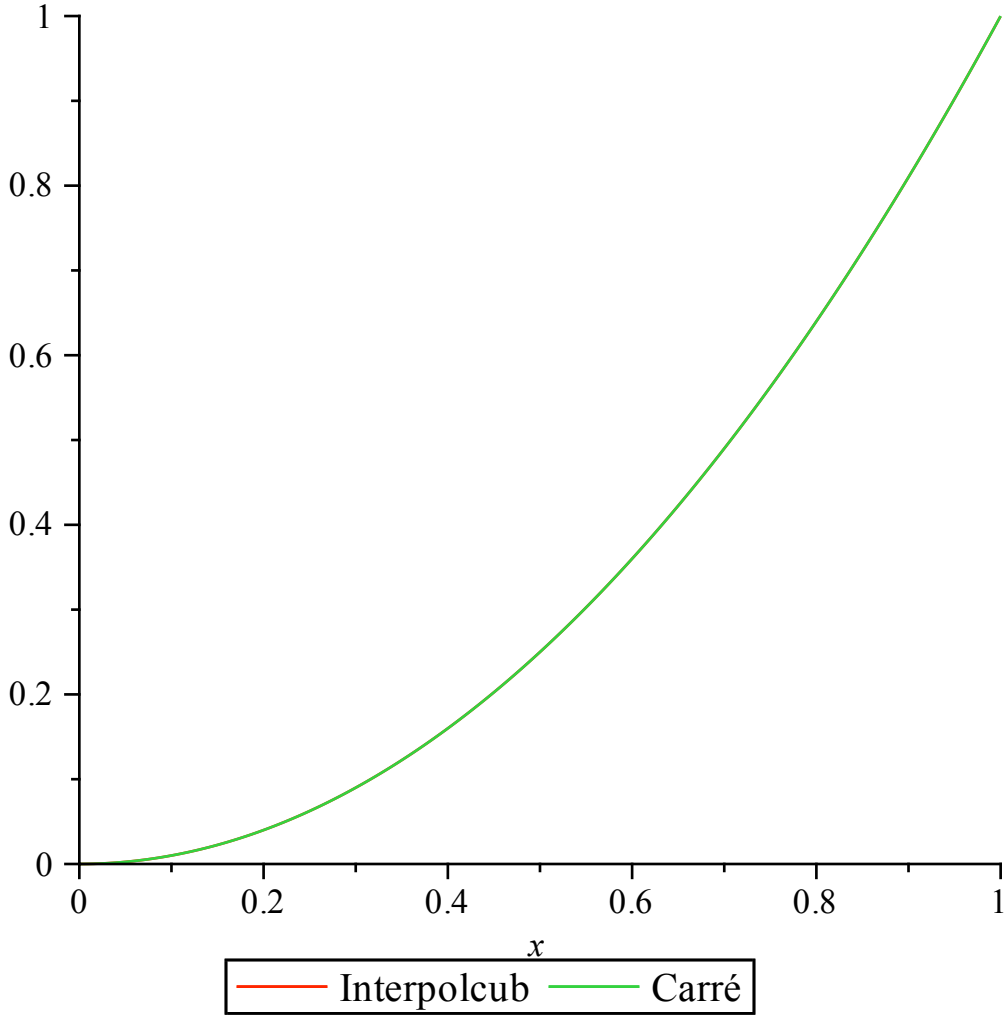

```
> f := interpolcub([-1, 0, 0.5, 1, 2], x):
  plot([f, x], x = 0..1, legend = ["Interpolcub", "Droite"]);
```



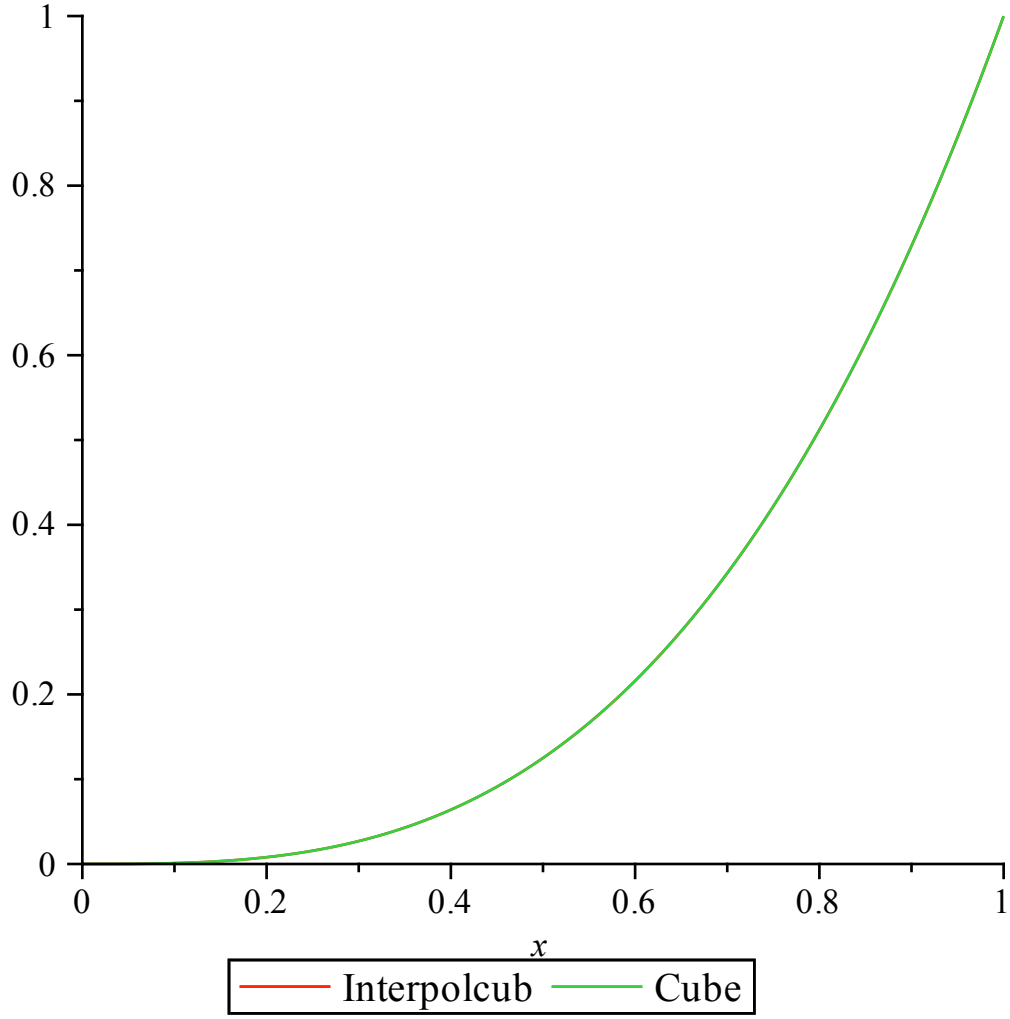

```
> f := interpolcub([-0.5, 0, 0.5, 1, 1.5], x):
  plot([f, x], x = 0..1, legend = ["Interpolcub", "Droite"]);
```



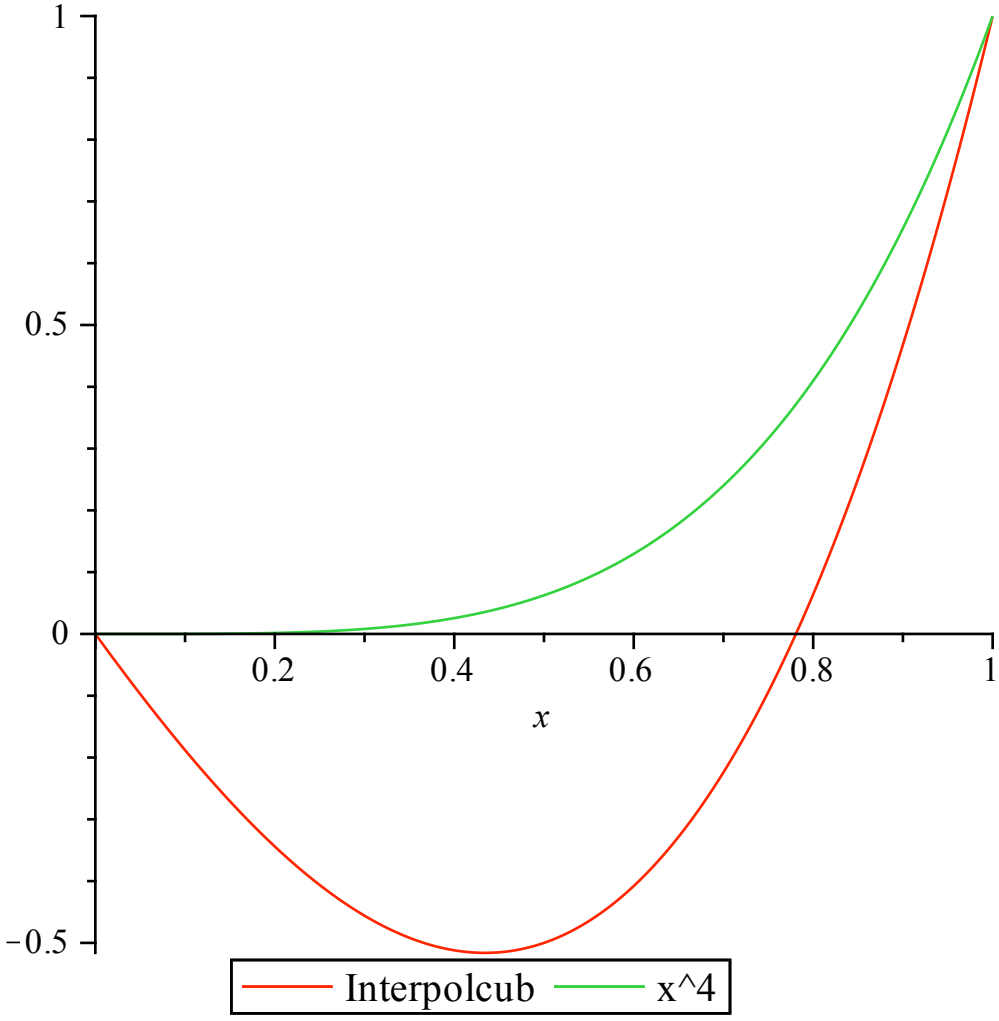

```
> f := interpolcub([1, 0, 1, 4], x) :
  plot([f, x^2], x = 0..1, legend = ["Interpolcub", "Carré"]);
```




```
> f := interpolcub([-1, 0, 1, 8], x):
  plot([f, x^3], x = 0..1, legend = ["Interpolcub", "Cube"]);
```




```
> f := interpolcub([1, 0, 1, 16], x):
  plot([f, x^4], x = 0..1, legend = ["Interpolcub", "x^4"]);
```



>

2.3.6 Exercices

Exercice 13

> $interpolcub2 := \mathbf{proc}(fX, x, a, b)$
> $interpolcub\left(fX, \frac{(x-a)}{b-a}\right):$
> **end proc**:

>