```
> restart
> with(LinearAlgebra) : with(ArrayTools) : with(plots) :
> # Natural Cubic Splines interpolation procedure;

CubicInterpolate := proc(f)
# HARDCODED ALERT
local segment := 0..1;
local h := 0.1;
local n := 10;
local i;
local xs := [seq(i, i = segment, h)];
local ys := [seq(f(i), i = segment, h)];
local matrixf := (i, j) -> if i = j then 4·h elif abs(i - j) = 1 then h else 0; end if;
local A := Matrix(n + 1, matrixf);
local fs := 6· Vector(n-1, j -> 1/h ((ys[j + 2] - ys[j + 1]) - (ys[j + 1] - ys[j])));
local a;
local b;
local c;
local d;
local S;
local calc_i;
A[1, 1] := 1;
A[1, 2] := 0;
A[n + 1, n + 1] := 1;
A[n + 1, n] := 0;
fs := Concatenate(1, 0, fs, 0);
c := LinearSolve(A, fs);
a := seq(ys[i], i = 2..n + 1);
d := seq((c[i] - c[i - 1])/h, i = 2..n + 1);
b := seq((ys[i] - ys[i - 1])/h + c[i]·h/3 + c[i - 1]·h/6, i = 2..n + 1);
S := (i, x) -> a[i] + b[i]·(x - xs[i + 1]) + c[i + 1]/2 ·(x - xs[i + 1])^2 + d[i]/6 ·(x - xs[i
    + 1])^3;
# HARDCODED ALERT
return x -> piecewise(
0 ≤ x < 0.1, S(1, x),
0.1 ≤ x < 0.2, S(2, x),
0.2 ≤ x < 0.3, S(3, x),
0.3 ≤ x < 0.4, S(4, x),
0.4 ≤ x < 0.5, S(5, x),
0.5 ≤ x < 0.6, S(6, x),
0.6 ≤ x < 0.7, S(7, x),
0.7 ≤ x < 0.8, S(8, x),
0.8 ≤ x < 0.9, S(9, x),
```

```
0.9 ≤ x ≤ 1, S(10, x)
);
end proc:
```

```
> # B-Splines interpolate procedure

BInterpolate := proc(f)
# HARDCODED ALERT
local segment := 0..1;
local h := 0.1;
local n := 12;
local eps := 10^(-9);
local i;
local xs := [-2·eps, -eps, seq(i, i = segment, h), 1 + eps, 1 + 2·eps];
local ys := [f(0), f(0), seq(f(i), i = segment, h), f(1), f(1)];
local lam := j→ piecewise(

j = 1, f(xs[1]),

1 < j < n, 1/2 (-f(xs[j + 1]) + 4 f((xs[j + 1] + xs[j + 2])/2) - f(xs[j + 2])),

j = n, f(xs[n + 1])

);
local B0 := (i, x) → piecewise(
xs[i] ≤ x < xs[i + 1], 1,
0
);
local B1 := (i, x) → (x - xs[i])/(xs[i + 1] - xs[i]) ·B0(i, x) + (xs[i + 2] - x)/(xs[i + 2] - xs[i + 1]) ·B0(i + 1, x);
local B2 := (i, x) → (x - xs[i])/(xs[i + 2] - xs[i]) ·B1(i, x) + (xs[i + 3] - x)/(xs[i + 3] - xs[i + 1]) ·B1(i + 1, x);
local P := x→sum(lam(i)·B2(i, x), i = 1..n);
return P;
end proc:
```

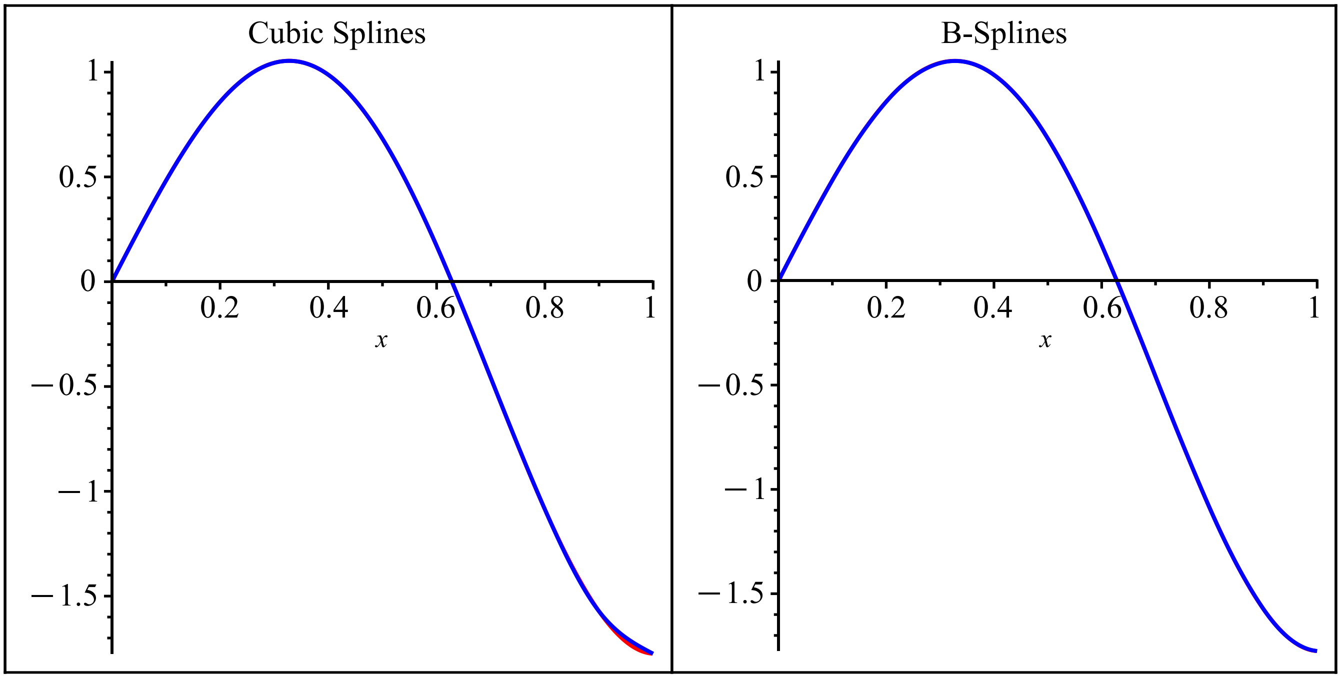
> # Проверка

```
> create_plots :=proc(f)
local plt := Array(1..2);
plt[1] := plot([f(x), CubicInterpolate(f)(x)], x = 0..1, color = [red, blue], title
    = "Cubic Splines");
plt[2] := plot([f(x), BInterpolate(f)(x)], x = 0..1, color = [red, blue], title = "B-Splines");
return plt;
end proc:
```

> $f := x \rightarrow \frac{\sin(5\,x)}{\cos(x)}$

$$f := x \mapsto \frac{\sin(5 \cdot x)}{\cos(x)} \tag{1}$$

> display(create_plots(f))



> # Сравнение с встроенными в Maple функциями

> with(CurveFitting) :

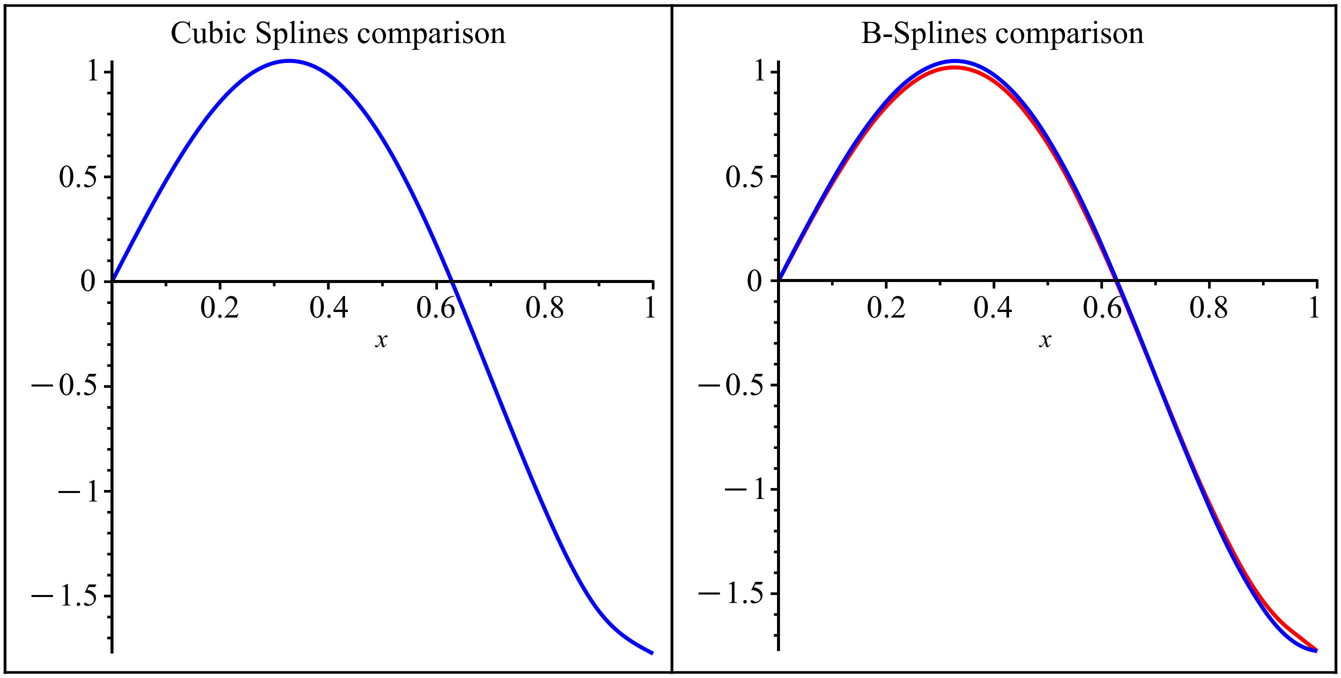
> $eps := 10^{-9}$

$$eps := \frac{1}{1000000000} \tag{2}$$

```
> comparison_plots :=proc(f)
  local plt := Array(1 ..2);
  local MapleCubicSpline := x→ Spline( [seq(i, i = 0 ..1, 0.1 ) ], [seq(f(i), i = 0 ..1, 0.1 ) ], x, degree
       = 3 );
  local MapleBSpline := x→BSplineCurve(
  [ −2·eps, −eps, seq(i, i = 0 ..1, 0.1 ), 1 + eps, 1 + 2·eps],
  [f(0), f(0), seq(f(i), i = 0 ..1, 0.1 ), f(1 ), f(1 ) ],
  x, order = 3 );
  plt[1 ] := plot( [MapleCubicSpline(x), CubicInterpolate(f) (x) ] , x = 0 ..1, color = [red, blue],
       title = "Cubic Splines comparison");
  plt[2 ] := plot( [MapleBSpline(x), BInterpolate(f) (x) ] , x = 0 ..1, color = [red, blue], title
       = "B-Splines comparison" );
  return plt;
  end proc:
```

Warning, (in MapleCubicSpline) `i` is implicitly declared local
Warning, (in MapleBSpline) `i` is implicitly declared local

> display(comparison_plots(f))



> # Б-Сплайны немного различаются. Вероятно, это связано с разной политипой выбора коэффицентов при B_j,2 (учитывая, что MapleBSpline строится только по узлам сетки, без требований к промежуточным значениям между ними, в отличии от используемого в самописном Б-Сплайне функционале)

> # Procedures to compute error of approximation for given function f

```
computeError := proc(f, interpolator)
local segment := 0 ..1;
local h := 0.01;
local i;
local xs := [seq(i, i = segment, h)];
local diff := x → abs(interpolator(x) − f(x));
local errors := map(diff, xs);
return max(errors);
end proc:

computeErrors := f → [computeError(f, CubicInterpolate(f)), computeError(f,
    BInterpolate(f))]:
```

> computeErrors(f)

$$[0.0207075785054398, 0.00131564602] \quad (3)$$

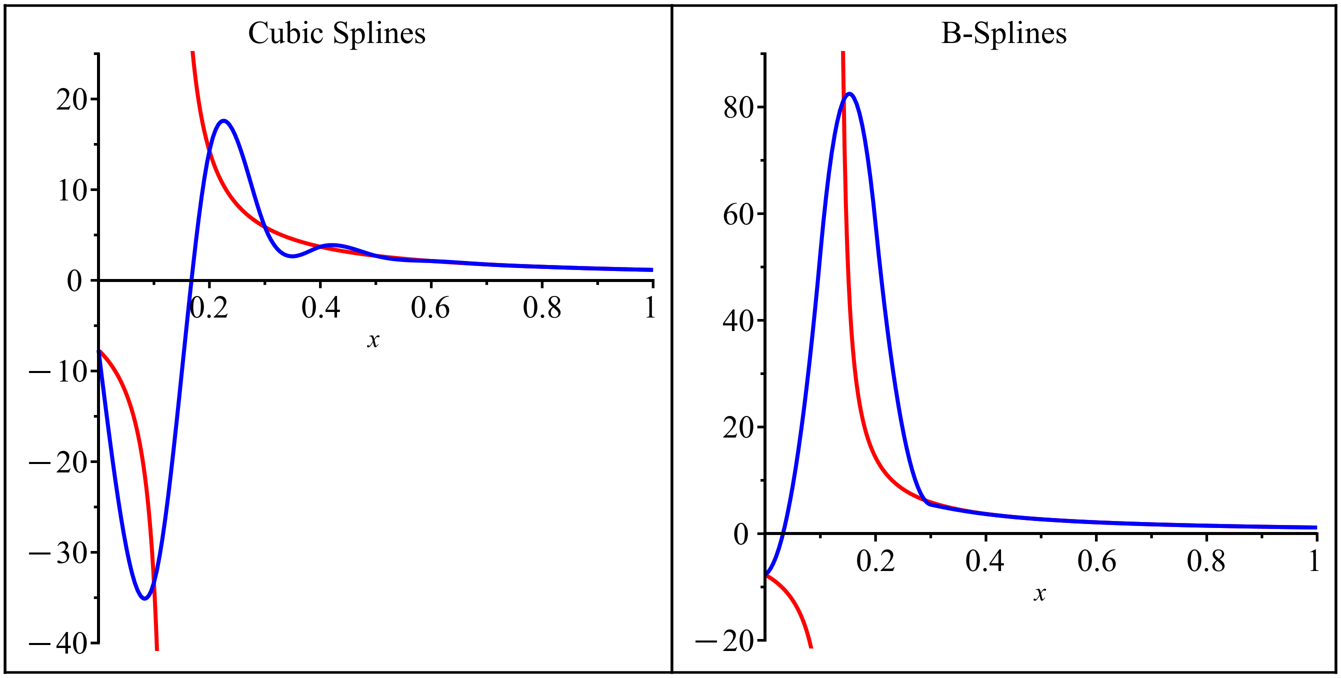
> # Тестовые функции

> # 0. Использовать сплайн аппроксимацию разумно только в том случае, если на функция пренадлежит $C^2[0, 1]$. В ином случае качество аппроксимации будет крайне низким.

> $f := x \rightarrow \frac{1}{x - 0.13}$

$$f := x \mapsto \frac{1}{x - 0.13} \tag{4}$$

```
> display(create_plots(f))
```




```
> computeErrors(f)
```

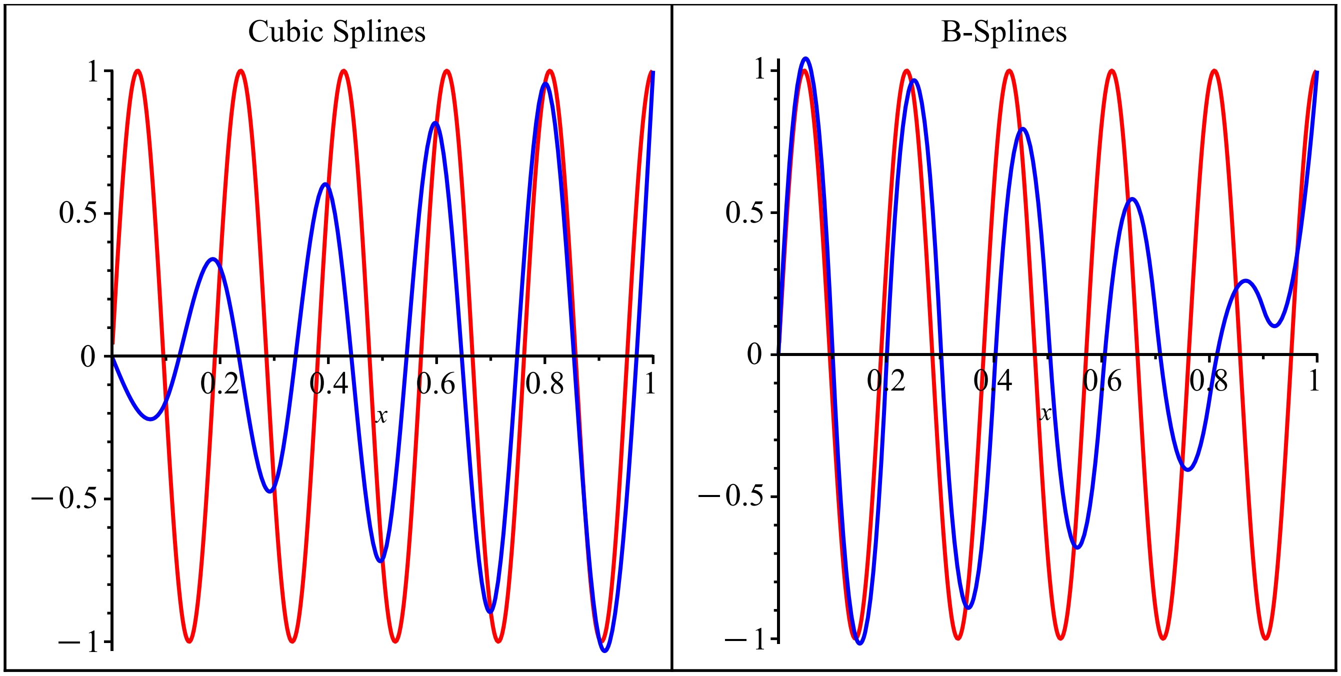
$$[\mathrm{Float}(\infty), \mathrm{Float}(\infty)] \tag{5}$$

> \# 1. В https://math.stackexchange.com/questions/2710326/what−kind−of−function−are−cubic−splines−not−good−at−approximating утверждается, что кубические сплайны плохо аппроксимируют "дерганные" на небольшом промежутке функции, вроде тригонометрических. Это обусловлено малой выборкой точек сетки по отношению к реальному поведению функции: в узлах сетки обе функции могут принимать одинаковое значение, при этом на остальном носителе сплайна значения сильно отличаются.

```
> f := x→sin(33 x)
```

$$f := x \mapsto \sin(33 \cdot x) \tag{6}$$

```
> display(create_plots(f))
```



> computeErrors(f)

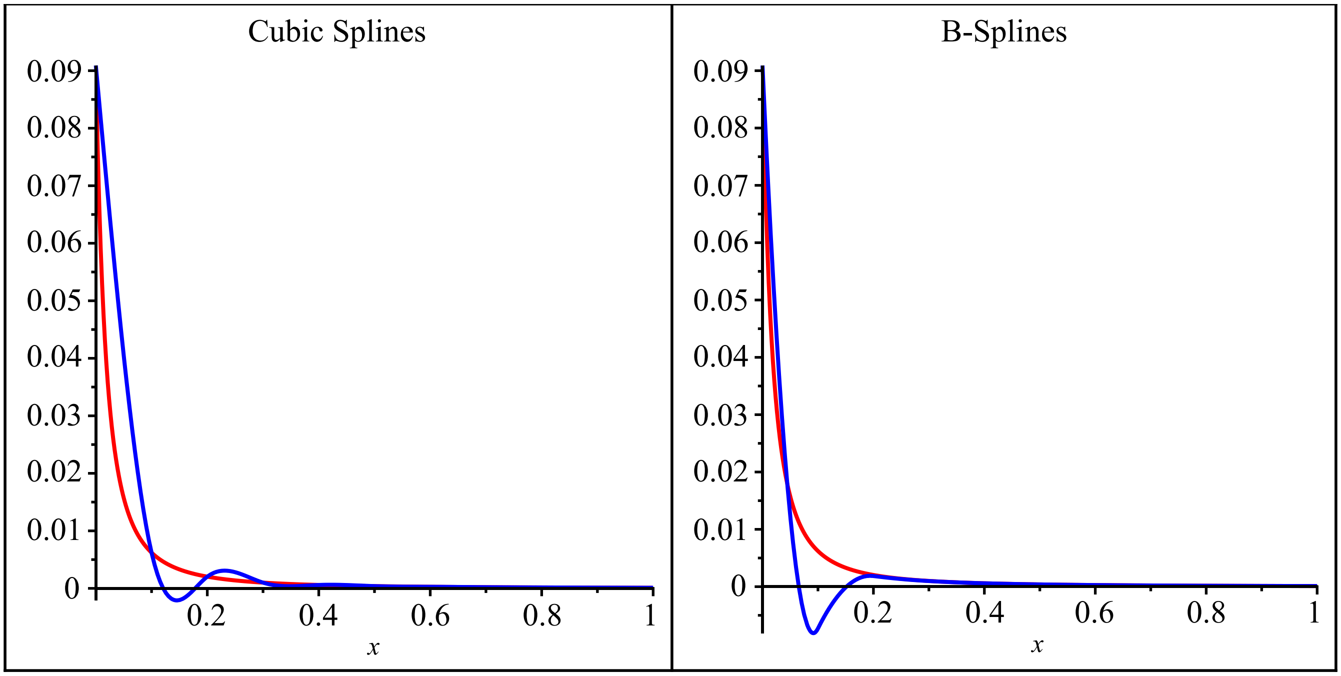
$$[1.19184786763525, 1.152259581] \tag{7}$$

> # 2. В https://drlvk.github.io/nm/section-drawbacks-spline-interpolation.html утверждается, что кубические сплайны не всегда сохраняют монотоность, даже если точки в выборке монотонны, и могут принимать отрицательные значения на положительных функциях.

> $f := x \rightarrow \dfrac{1}{1 + 10(30x + 1)^2}$

$$f := x \mapsto \frac{1}{1 + 10 \cdot (30 \cdot x + 1)^2} \tag{8}$$

> display(create_plots(f))



> computeErrors(f)

[0.0327136803915917, 0.01532634738] **(9)**

> #` `Как в*идно, функция f монотонна на [0,1] и принимает только положительные значения. Однако, как и в предыдущем случае, недостаток узлов приводит к дефектам, вроде нарушения монотонности и появлению отрицательных значений, даже если природа задачи их не подразумевает.*