```
> restart;
# Cubic −splain
> n := 10 :;
  h := 1/n :;
> xc := Array(0 ..n, i→i·h) :;
> eqs := [cc[0] = 0, cc[n] = 0] :;
  for ic from 1 to n − 1 do
   eqs := [op(eqs), cc[ic − 1]·h + 4·h·cc[ic] + cc[ic + 1]·h = 6
       ·((f(xc[ic + 1]) − f(xc[ic]))/h − (f(xc[ic]) − f(xc[ic − 1]))/h)];
  end do:;
  assign(fsolve(eqs)) :;
> ac := Array(1 ..n, i→f(xc[i])) :;
  bc := Array(1 ..n, i→(f(xc[i]) − f(xc[i − 1]))/h + cc[i]·h/3 + cc[i − 1]·h/6) :;
  dc := Array(1 ..n, i→(cc[i] − cc[i − 1])/h) :;
> sc(x, i) := ac[i] + bc[i]·(x − xc[i]) + cc[i]/2·(x − xc[i])^2 + dc[i]/6·(x
      − xc[i])^3 :;
> Cubic :=proc(x,f)
   local i;
   for i from 1 to n do
    if x ≥ xc[i − 1] and x ≤ xc[i] then
     return sc(x, i);
    end if;
   end do;
  end proc:
> Sc(x) := Cubic(x,f) :;
# B-splain
> eps := 10^(−8) :;
> xb := [−2·eps,−eps, seq(i·h, i=0 ..n), 1 + eps, 1 + 2·eps] :;
  yb := [f(0), f(0), seq(f(i·h), i=0 ..n), f(1), f(1)] :;
> ab(i) := piecewise(

  i=1, yb[1],
```

$$1 < i < n+2,\ \frac{1}{2}\left(-yb[i+1] + 4\cdot f\left(\frac{xb[i+1]+xb[i+2]}{2}\right) - yb[i+2]\right),$$

$$i = n+2,\ yb[n+3]$$

$$\Big) :;$$

> $B[0](i,x) := piecewise(xb[i] \le x < xb[i+1], 1, 0) :;$

$$B[1](i,x) := \frac{x - xb[i]}{xb[i+1]-xb[i]}\cdot B[0](i,x) + \frac{xb[i+2]-x}{xb[i+2]-xb[i+1]}\cdot B[0](i+1,x) :;$$

$$B[2](i,x) := \frac{x - xb[i]}{xb[i+2]-xb[i]}\cdot B[1](i,x) + \frac{xb[i+3]-x}{xb[i+3]-xb[i+1]}\cdot B[1](i+1,x) :;$$

> $BSplane(x) := sum(ab(i)\cdot B[2](i,x), i = 1..n+2) :;$

> $Sb(x) := BSplane(x) :;$

> $with(CurveFitting) :;$

> $MapleCubic(x) := Spline([seq(i, i = 0..1, 0.1)], [seq(f(i), i = 0..1, 0.1)], x, degree = 3) :;$

Warning, (in MapleCubic) `i` is implicitly declared local

> $\mathrm{MapleBSpline}(x) := BSplineCurve($
> $[-2\cdot eps, -eps, seq(i, i = 0..1, 0.1), 1 + eps, 1 + 2\cdot eps],$
> $[f(0), f(0), seq(f(i), i = 0..1, 0.1), f(1), f(1)],$
> $x, order = 3) :;$

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

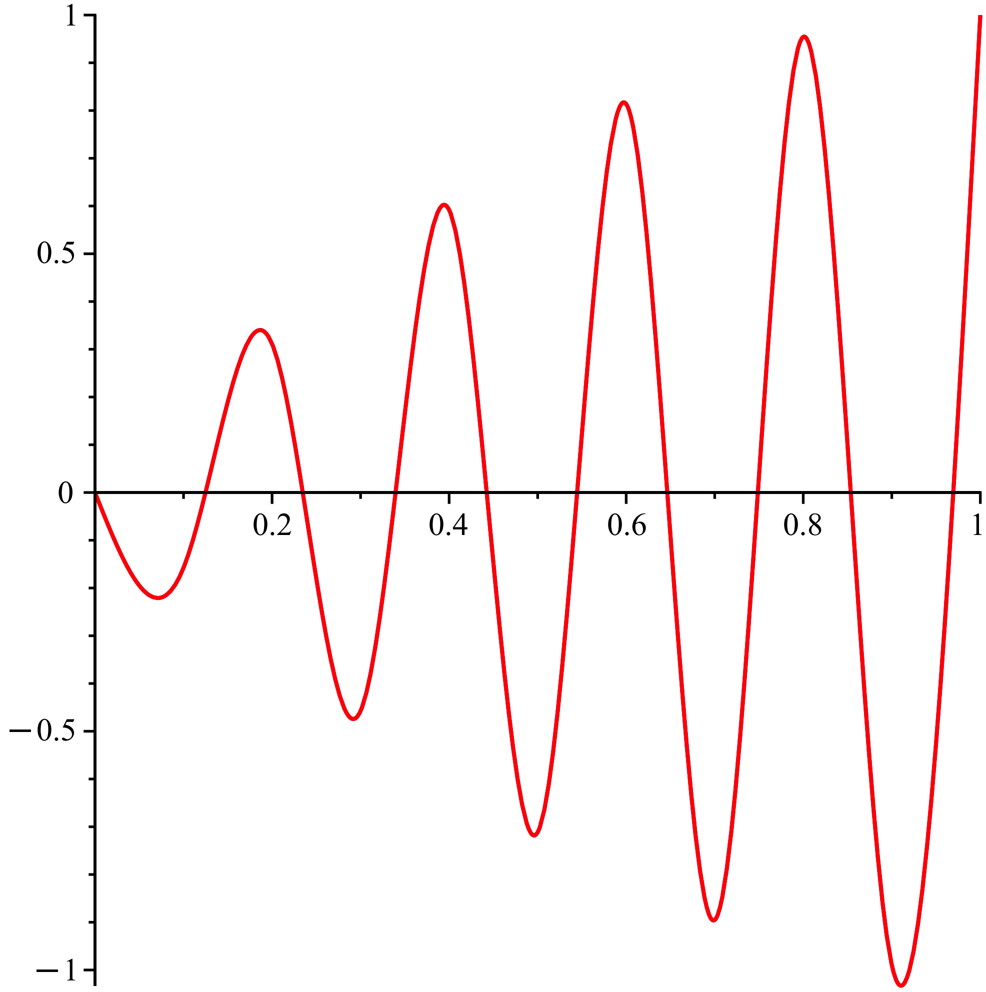
```
> # Procedures to compute error of approximation for given function f

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 evalf(max(errors));
end proc:
```

```
> computeErrors := f → [evalf(computeError(f, Sc)) ,
      evalf(computeError(f, Sb)) ] :
```

 # Сравним полученные реализации сплайнов с реализациями Maple

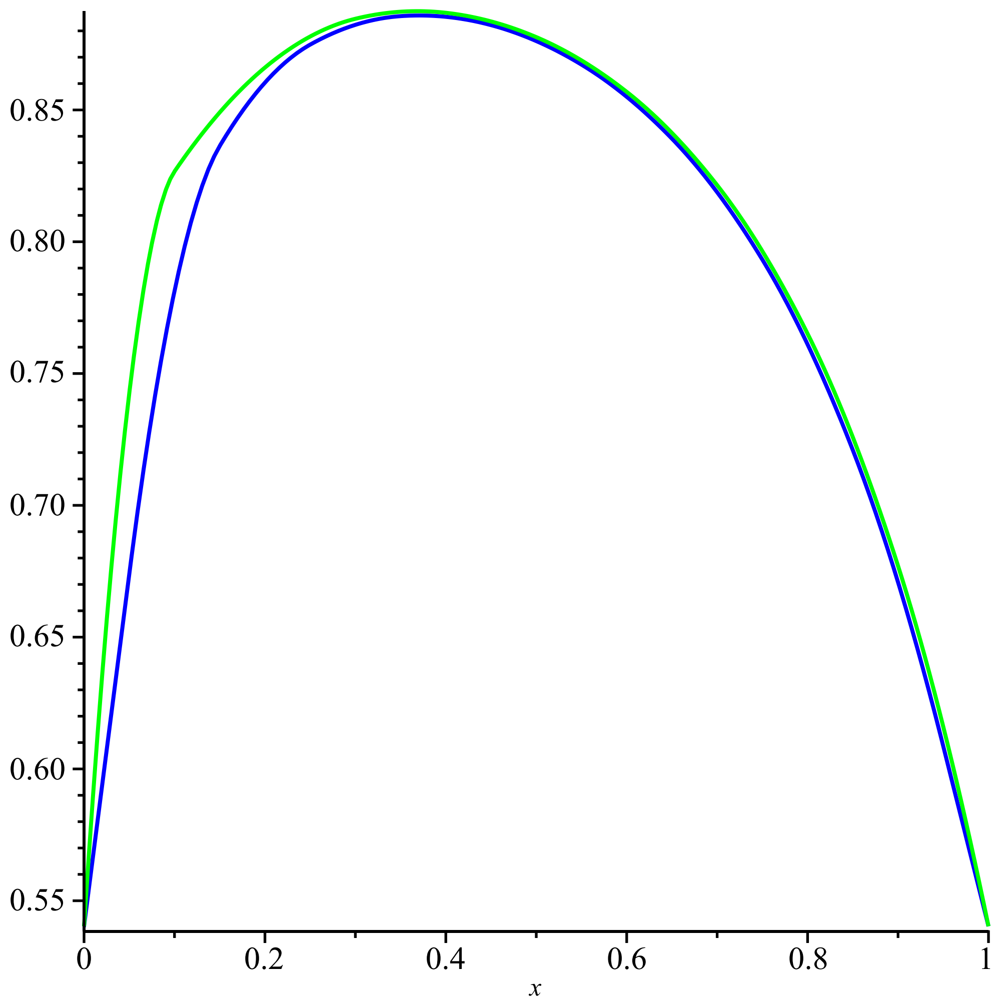
> $f(x) := \sin(33\cdot x) :;$

> $plot([MapleCubic, Sc], 0..1, color=[blue, red]);$



> $f(x) := \cos\left(x^{2\cdot x}\right)$:;

> $plot([MapleBSpline(x), Sb(x)], x=0..1, color=[blue, green])$;

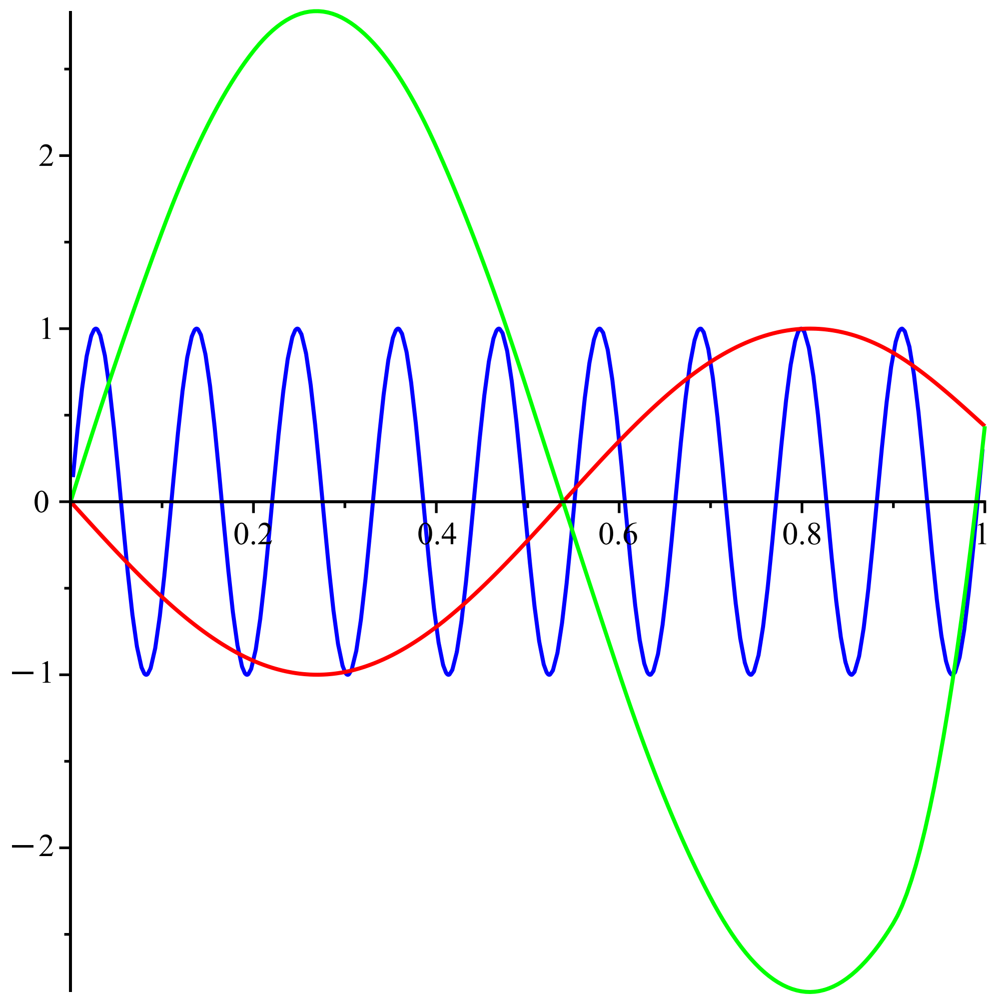


 # Первый график подтверждает корректность реализации кубисечкого сплайна, на втором же можно увидеть некоторую разницу, вероятно обусловленную разным выбром коэфицентов.

 # Покажем, что с высокочастоной переодической функцией оба сплайна несовсем соответсвуют действительности, потому что коэфиценты не успевают реагировать на постоянно меняющиеся скочки функции .

> $f(x) := \sin(57 \cdot x)$:;

> $plot([f, Sc, Sb], 0..1, color = [blue, red, green])$;



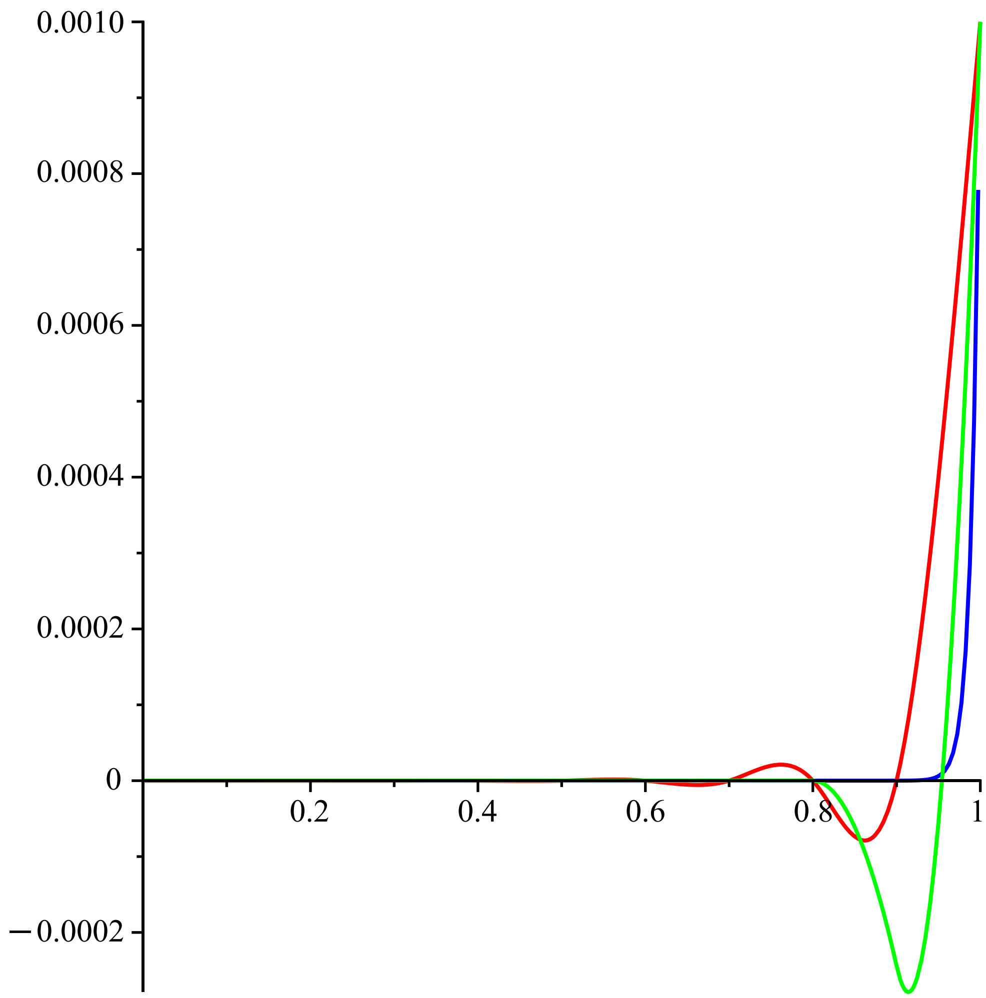
> $computeErrors(f)$;

$$[1.986956021, 3.828714991] \tag{1}$$

 # Далее возьмём функциию 100 й степени от x, и посмотрим на приближения полученные сплайнами . Результат какжется предсказуемым, ведь продифференцировав функции производные будут вести себя совершенно по − разному . Кроме того рассуждения об апроксимации таких функций можно посмотреть здесь https ://ru.wikipedia.org/wiki /%D0%A4%D0%B5%D0%BD%D0%BE%D0%BC%D0%B5%D0%BD_%D0%A0\ %D1%83%D0%BD%D0%B3%D0%B5 .

> $f(x) := \dfrac{(10\,x^{100})}{10000}$:;

> $plot([f, Sc, Sb], 0 ..1, color = [blue, red, green])$;



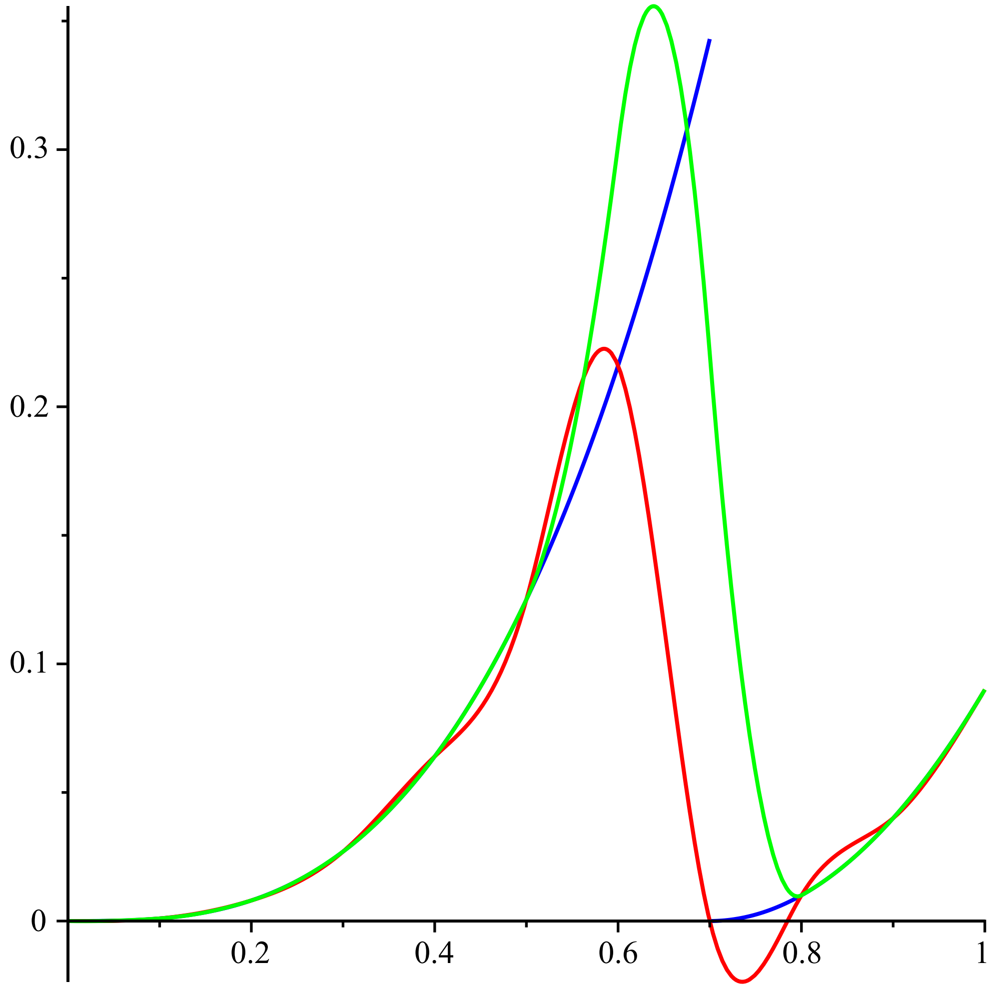
> $computeErrors(f)$;

$$[0.0006159426107, 0.0003536556115] \quad \textbf{(2)}$$

 # Далее возьмём функцию заданную с разрывом, однако котрую можно определить до гладкой, таким образом, получив резкий скачок. Из-за чего B-сплайн отреагирует на этот скачок лучше, в отличие от кубического.

> $f(x) := piecewise\left(x < 0.7,\ x^3,\ (x - 0.7)^2\right)$:;

> $plot([f, Sc, Sb], 0..1, color = [blue, red, green])$;



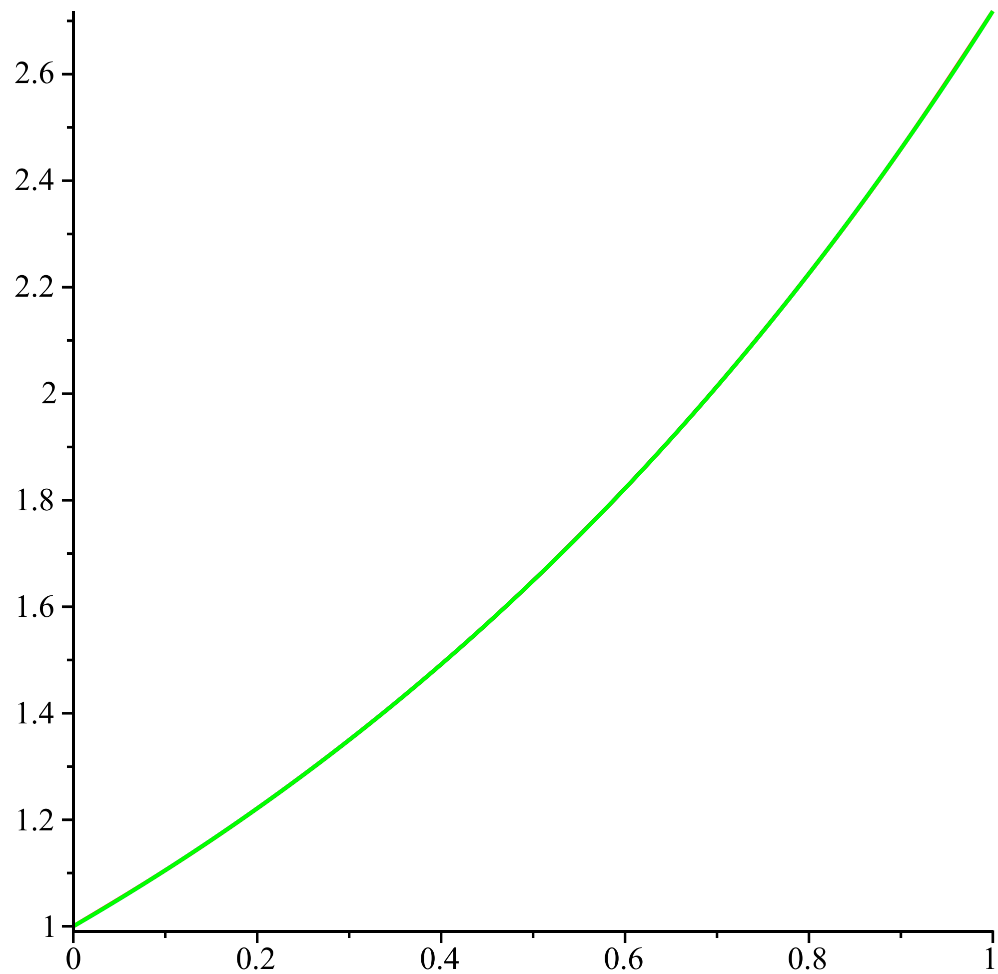
> *computeErrors*(*f*) ;

[0.3113484712, 0.2206250000] **(3)**

 # Также в качестве функции для апроксимации рассмотрим экспоненту, судя по графику ниже оба сплайна достаточно точно апроксимируют эту простую функцию, Однако сравнив ошибки можно заметить, что B-сплайн справился с этой задачей лучше.

> $f(x) := \exp(x)$:;

> *plot*([*f*, *Sc*, *Sb*], 0 ..1, *color* = [*blue*, *red*, *green*]);
computeErrors(*f*);



[0.001330089799, 0.000022996] **(4)**

>

>

>