```
> restart;
```

\# Построим кубический сплайн

```
> 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) :;
```

\# Построим B-сплайн

```
> 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] ,
```

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

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

$)$:;

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

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

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

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

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

> $with(CurveFitting)$:;

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

```
> 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) :;
```

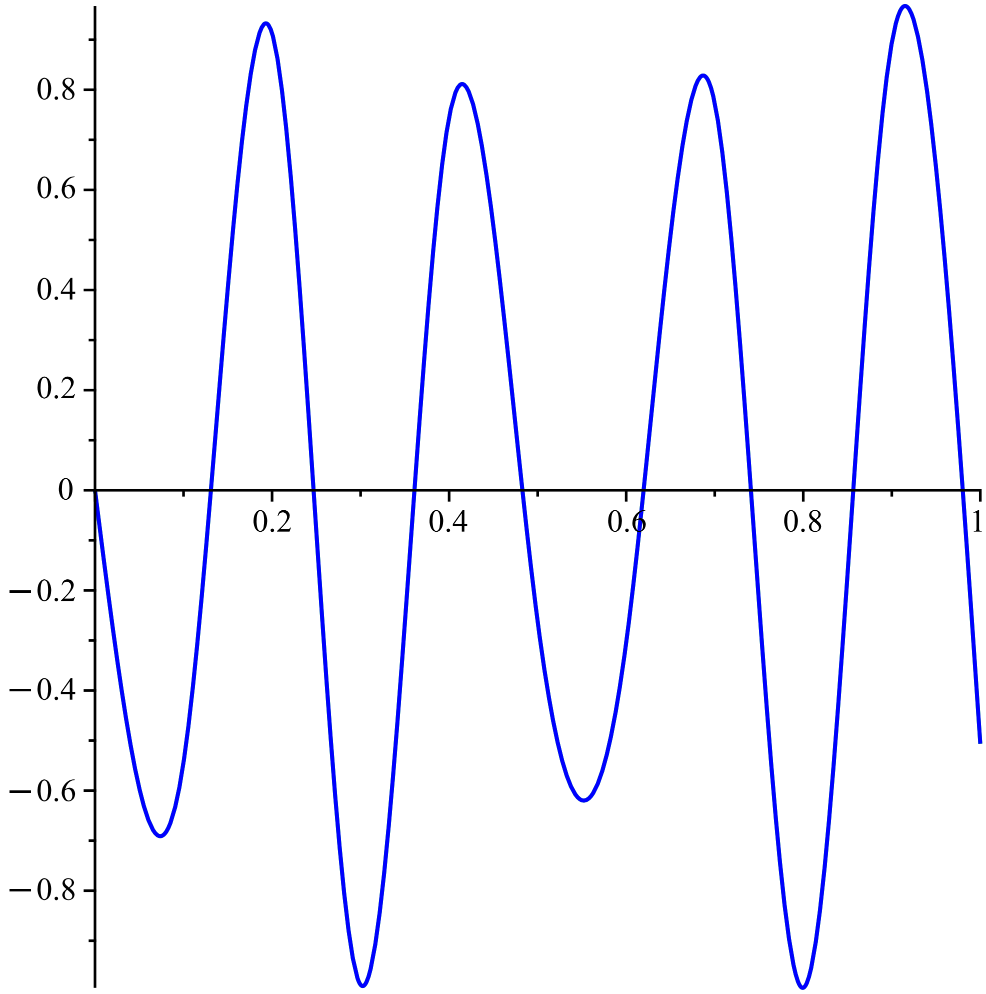
```
>
    # Процедуры вычисления погрешности аппроксимации для заданной
    функции f

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

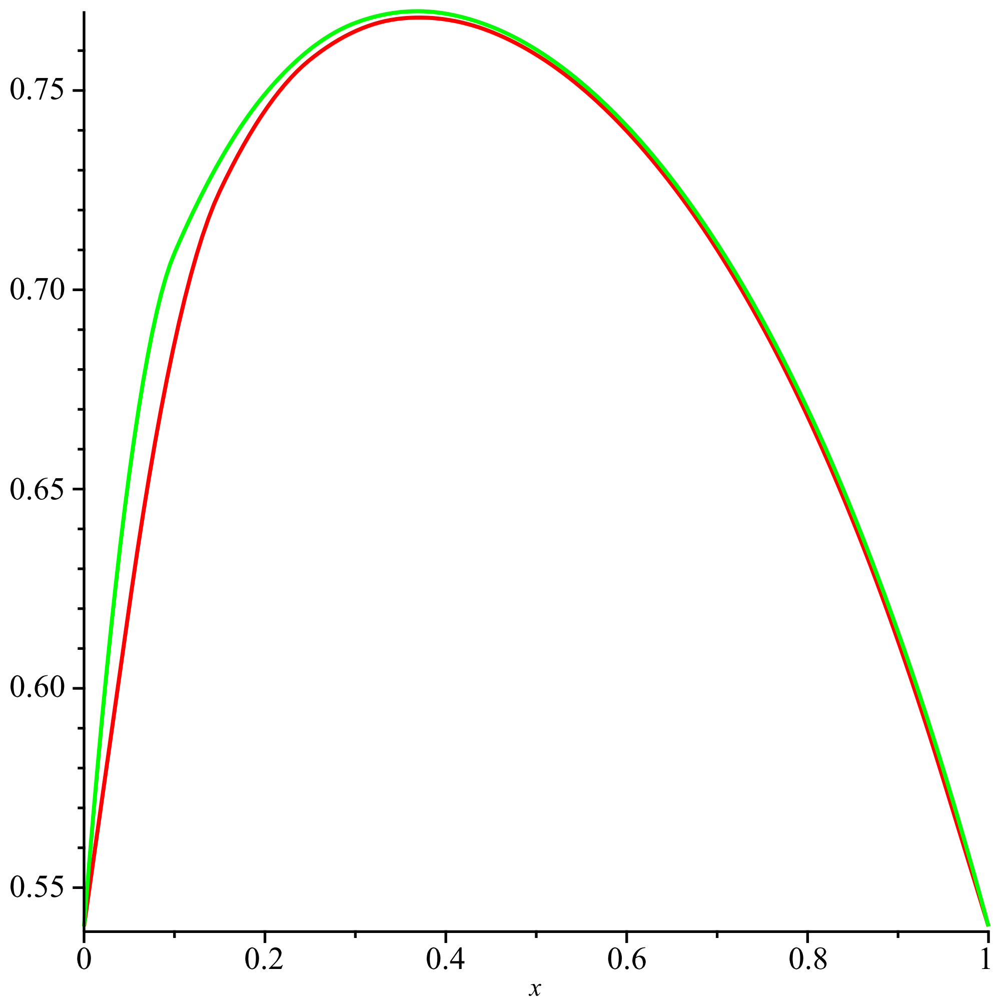
> $f(x) := \sin(100 \cdot x)$:;

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



> $f(x) := \cos(x^x)$:;

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

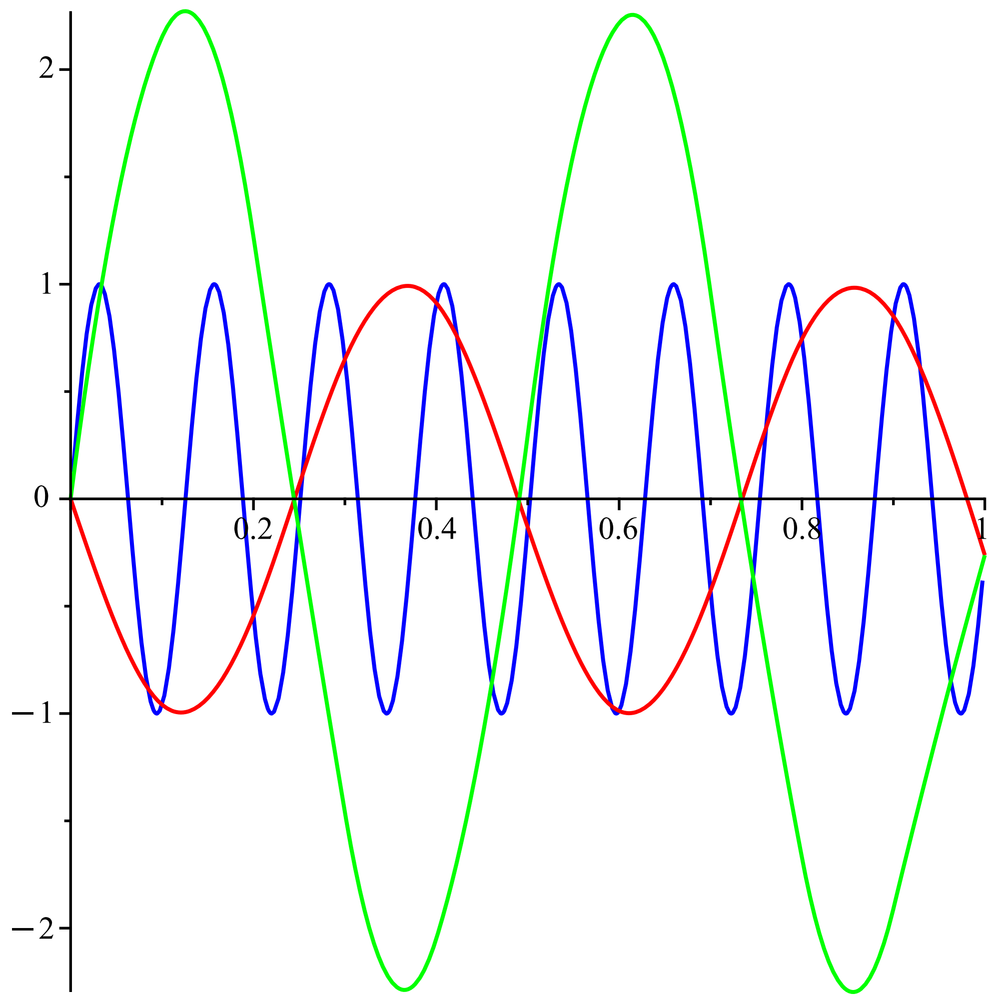


 # Первый график подтверждает корректность реализации кубисечкого сплайна, на втором же можно увидеть некоторую разницу, вызванную разным выбром коэфицентов.

 # С высокочастоной переодической функцией оба сплайна не совсем соответсвуют действительности, потому что коэфиценты не успевают реагировать на постоянно меняющиеся скочки. Разница обусловлена тем, что с [i] в стандартной библиотеке отличаются от тех, что выбрали мы. Однако, и то, и то -- B-сплайн.

> $f(x) := \sin(50 \cdot x)$:;

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



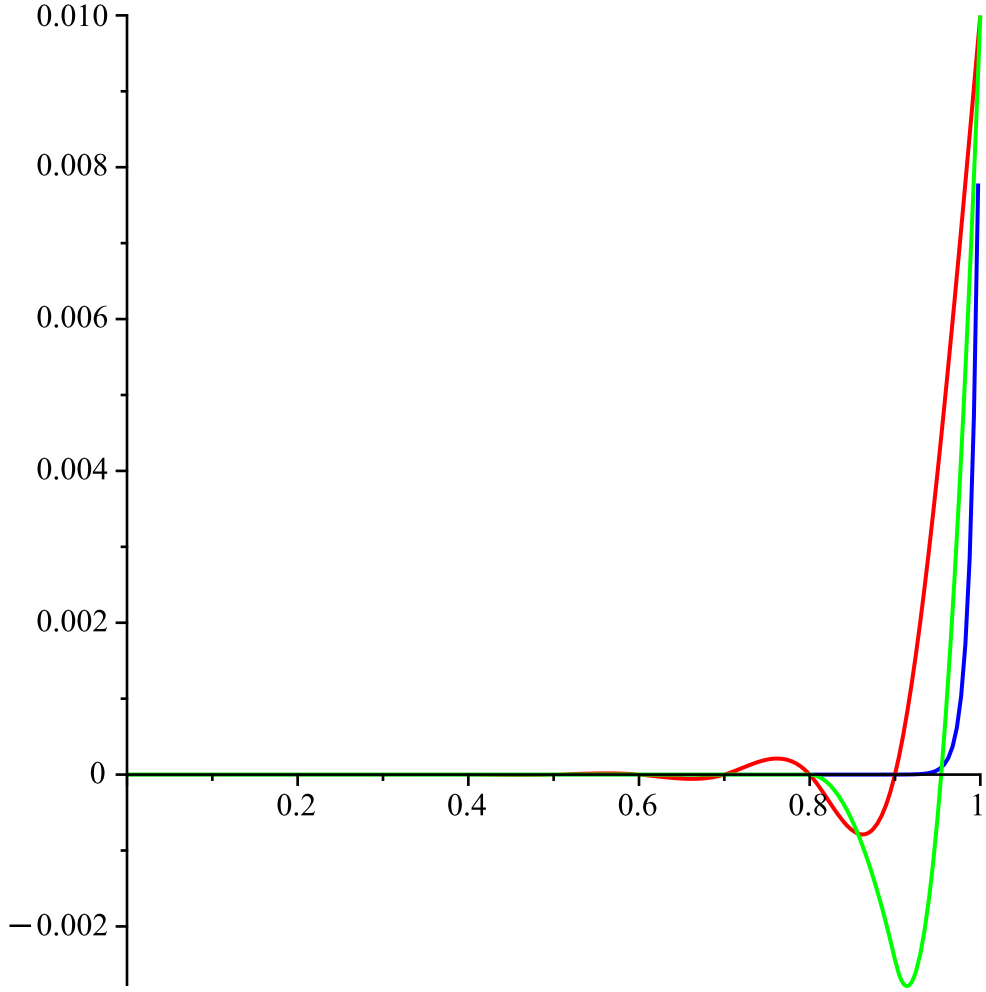
> $computeErrors(f)$;

$$[1.975780066, 3.205291495] \qquad (1)$$

 # Возьмём функциию 100 й степени от x,
и посмотрим на приближения полученные сплайнами
. Результат оказался предсказуемым,
ведь продифференцировав функции производные будут вести себя совершенно поразному

> $f(x) := \frac{(10\,x^{100})}{1000}$:;

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



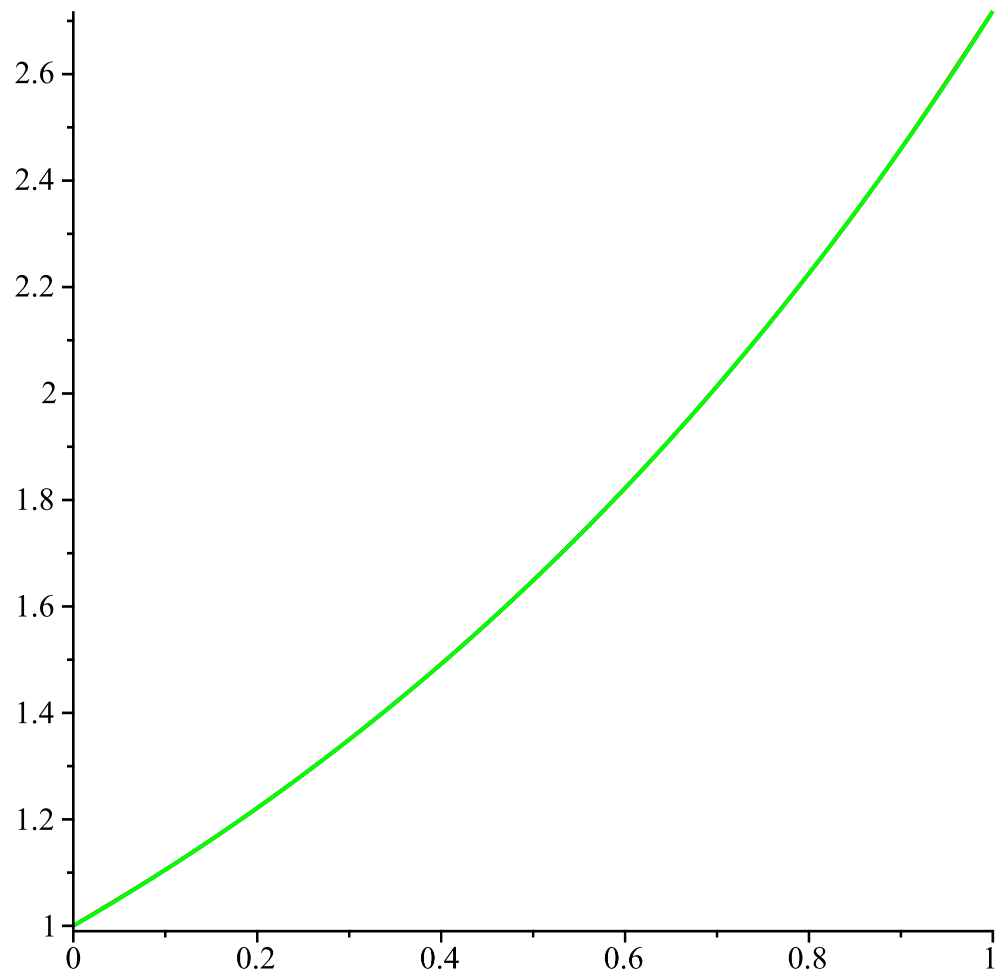
> *computeErrors*(*f*) ;

$$[0.006159426107, 0.003536556115] \tag{2}$$

 # Оба сплайна достаточно точно апроксимируют экспоненту, но 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] (3)

>

>