$restart$

$f := x \to \sin(6 \cdot x);$

$$f := x \mapsto \sin(6 \cdot x) \tag{1}$$

$n := 10 : h := \frac{1}{n} :$

$helper := k \to k \cdot h : t := Array(0..n, helper)$

$$t := \left[0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1, \cdots \text{0 .. 10 Array}\right] \tag{2}$$

Cubic spline

// реализация взята с https://fncbook.github.io/fnc/localapprox/splines.html

$with(LinearAlgebra) :$

$ai := Vector(n, symbol = a) : \; bi := Vector(n, symbol = b) : \; ci := Vector(n, symbol = c) : \; di := Vector(n, symbol = d) :$

$im := IdentityMatrix(n) : \; m := \langle ai, \; bi, ci, di \rangle \; :;$

$helper := \; i \to f(t[i - 1]) : \; yi \; := Vector(n + 1, \; helper) :$

$Zn := ZeroMatrix(n) : \; H := DiagonalMatrix(\; Vector(n, h) \;)$

$$H := \begin{bmatrix} \frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{10} \end{bmatrix} \tag{1.1}$$

$eq\_leftc := \langle im \mid Zn \mid Zn \mid Zn \rangle \; m = Matrix(\, yi[1 .. n]) :$
$buf := Vector(n, h) : H := DiagonalMatrix(\mathrm{buf}) :$
$eq\_rightc := \langle im \mid H \mid H^2 \mid H^3 \rangle \; m = Matrix(yi[2 .. n+1]) :$
$helper := (i, j) \rightarrow$ **if** $i = j$ **then** 1 **else if** $i + 1 = j$ **then** -1 **else** 0 **end end**:; $J := Matrix(n, helper)$

$$J := \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$helper := (i, j) \rightarrow$ **if** $i = j$ **then** 1 **else** 0 **end**: $E := Matrix(n-1, n, helper)$

$$E := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.3}$$

$eq\_d1 := E < Zn \mid J \mid 2\, H \mid 3\, H^2 > m = Matrix(\ Vector(n-1, 0)\ ):$
$eq\_d2 := E < Zn \mid Zn \mid J \mid 3\, H > m = Matrix(\ Vector(n-1, 0)\ ):$
$eq\_d2\_1 := \ ci[1] + 3\, di[1] = 0 :$
$eq\_d2\_n := \ ci[n] + 3\, di[n] = 0 :$

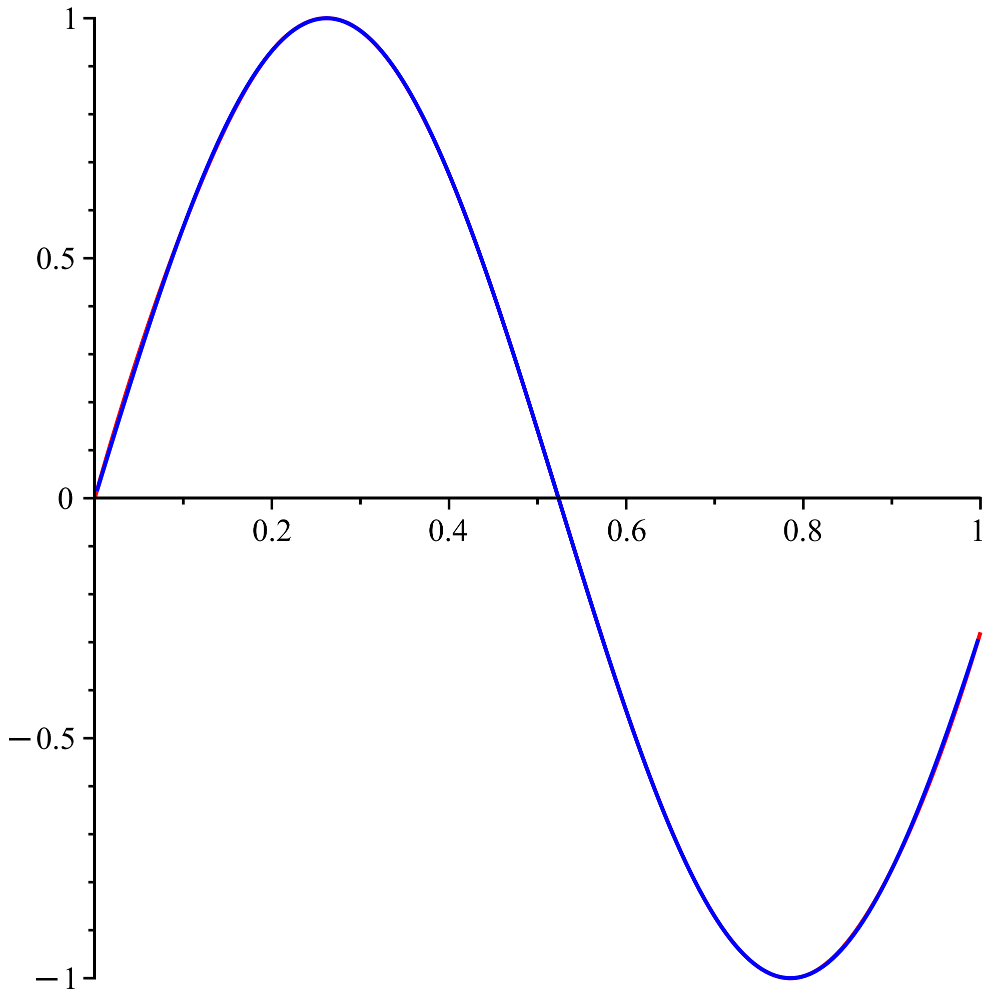
$buf := solve(\{eq\_d2\_1, eq\_d2\_n\}$ **union** $convert(\ eq\_d1, setofequations)$ **union** $convert(\ eq\_d2, setofequations)$ **union** $convert(\ eq\_rightc, setofequations)$ **union** $convert(\ eq\_leftc, setofequations)\ \ ) :;$
$ai := eval(ai, buf) :; bi := eval(bi, buf) : ci := eval(ci, buf) : di := eval(di, buf) :$

$helper := (i, x) \rightarrow ai[i] + bi[i]\,(\,x - t[i-1\,])+ ci[i]\,(\,x - t[i-1\,])^2 + di[i]\,(\,x - t[i-1\,])^3 :$

$$Cs := x \rightarrow helper\left(\min\left(\mathrm{floor}\left(\frac{x}{h}\right) + 1, n\right), x\right);$$

$$Cs := x \mapsto helper\left(\min\left(\left\lfloor \frac{x}{h} \right\rfloor + 1, n\right), x\right) \tag{1.4}$$

$plot([Cs, f], 0 ..1, color = [red, blue])$



Procedure

$Cubic\_spline := \mathbf{proc}(f)$

$\mathbf{local}\ buf, ai, bi, ci, di, im, m, yi, Zn, helper, H, J, eq\_leftc, eq\_rightc, E, eq\_d1, eq\_d2, eq\_d2\_1, eq\_d2\_n, Cs;$

$\mathbf{local}\ a, b, c, d;$

$ai := Vector(n, symbol = a) ::;\ bi := Vector(n, symbol = b) ::;\ ci := Vector(n, symbol = c) ::;\ di := Vector(n, symbol = d) ::;$

$im := IdentityMatrix(n) ::;\ m := \langle ai, bi, ci, di \rangle ::;$

$helper := i \rightarrow f(t[i-1]) ::;\ yi := Vector(n+1, helper) ::;$

$Zn := ZeroMatrix(n) ::;\ H := DiagonalMatrix(Vector(n, h)) ::;$

```
eq_leftc := Typesetting[delayDotProduct](⟨im | Zn | Zn | Zn⟩, m, true) = Matrix(yi[1 .. n]);
buf := Vector(n, h);
H := DiagonalMatrix(buf);
eq_rightc := Typesetting[delayDotProduct](⟨im | H | H^2 | H^3⟩, m, true) = Matrix(yi[2 .. n + 1]);
helper := (i, j) → if i = j then 1; else if i + 1 = j then −1; else 0; end if; end if;
J := Matrix(n, helper);

helper := (i, j) → if i = j then 1; else 0; end if;
E := Matrix(n − 1, n, helper);
eq_d1 := Typesetting[delayDotProduct](Typesetting[delayDotProduct](E, ⟨Zn | J | 2 * H | 3 * H^2⟩,
    true), m, true) = Matrix(Vector(n − 1, 0));
eq_d2 := Typesetting[delayDotProduct](Typesetting[delayDotProduct](E, ⟨Zn | Zn | J | 3 * H⟩, true),
    m, true) = Matrix(Vector(n − 1, 0));
eq_d2_1 := ci[1] + 3 * di[1] = 0;
eq_d2_n := ci[n] + 3 * di[n] = 0;


buf := solve({eq_d2_1, eq_d2_n} union convert( eq_d1, setofequations) union convert( eq_d2,
    setofequations) union convert( eq_rightc, setofequations) union convert( eq_leftc,
    setofequations) ) :;
ai := eval(ai, buf) :; bi := eval(bi, buf) :; ci := eval(ci, buf) :; di := eval(di, buf) :;
helper := (i, x) → ai[i] + bi[i] (x − t[i − 1]) + ci[i] (x − t[i − 1])^2 + di[i] (x − t[i − 1])^3 :;
    Cs := x → helper(min(floor(x/h) + 1, n), x);
return Cs
end:
```

B−splines

// реализация взята с https://fncbook.github.io/fnc/localapprox/splines.html

```
with(LinearAlgebra) :

epsilon := 10^(−7) :
maxd := 2 : # степень сплайна

xi := i → if i < 0 then t[0] + epsilon·(−i) else if i > n then t[n] + epsilon·(i − n) else t[i] end end:

B0 := j → (x→ if x ≥ xi(j) and x < xi(j + 1) then 1 else 0 end) :
```

$BD := (j, d) \rightarrow \left(x \rightarrow \frac{x - \mathrm{xi}(j)}{\mathrm{xi}(j+d) - \mathrm{xi}(j)} \cdot B[j,\ d-1\ ](x) + \frac{\mathrm{xi}(j+d+1) - x}{\mathrm{xi}(j+d+1) - \mathrm{xi}(j+1)} \cdot B[j+1,\ d-1\ ](x)\right)$:;

create_B := **proc**(*di*) **local** *j*; **global** *B*; **for** *j* **from** −*maxd*− *maxd* + *di* **to** *n* + 2· *maxd*− *di* **do** **if** *di* = 0 **then** *B*[*j*, *di*] := *B0*(*j*) **else** *B*[*j*, *di*] := *BD*(*j*, *di*) **end** **end** **end**:
for *i* **from** 0 **to** *maxd* **do** *create_B*(*i*) **end**:

sum_helper := **proc**(*fs*, *a*, *b*) **local** *i*, *s*; *s* := 0; **for** *i* **from** *a* **to** *b* **do** *s* := *s* + *fs*(*i*) **end**; **return** *s* **end**:

 # коэффиценты с пары. По микро наблюдениям работают хуже чем из книжки, но иногда получается лучше + универсальные - подходят для любых степеней сплайна

$xistar := j \rightarrow \frac{sum\_helper(\mathrm{xi},\ (j+1), (j+maxd))}{maxd}$:

use_lesson_c := **proc**(*f*)**global** *ci*; *ci* := *i*→ *f*(*xistar*(*i*)) **end**:

 # коэффиценты из https://www.uio.no/studier/emner/matnat/math/MAT4170/v18/pensumliste/splinebook-2018.pdf

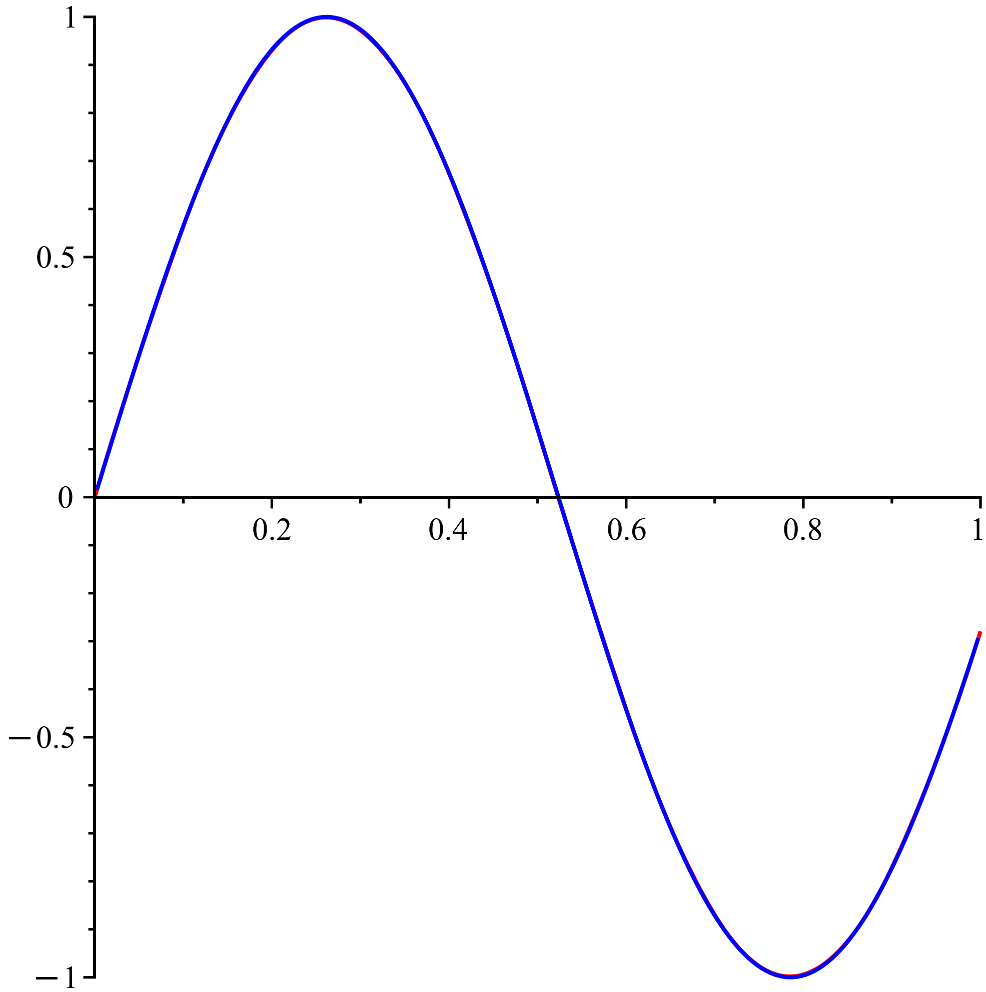
use_book_c := **proc**(*f*)**global** *ci*; *ci* := *i*→
if *i* ≤ −*maxd* **then** *f*(xi(0)) **else**
if *i* ≥ *n* **then** *f*(xi(*n*)) **else**
$\frac{1}{2} \cdot \left(-f(\mathrm{xi}(i+1)) + 4 \cdot f\left(\frac{\mathrm{xi}(i+1) + \mathrm{xi}(i+2)}{2}\right) - f(\mathrm{xi}(i+2))\right)$
end **end**
end:
----
if *maxd* = 2 **then** *use_book_c*(*f*) **else** *use_lesson_c*(*f*) **end**

$$i \mapsto \mathbf{if}\ i \le -maxd\ \mathbf{then}\ f(\xi(0))\ \mathbf{else\ if}\ n \le i\ \mathbf{then}\ f(\xi(n))\ \mathbf{else}\ -\frac{f(\xi(i+1))}{2} + 2 \cdot f\left(\frac{\xi(i+1)}{2} + \frac{\xi(i+2)}{2}\right) - \frac{f(\xi(i+2))}{2}\ \mathbf{end\ if\ end\ if} \tag{2.1}$$

$helper := x \rightarrow (j \rightarrow ci(j) \cdot B[j, \mathrm{maxd}](x))$:
$Bs := x \rightarrow sum\_helper(helper(x)\ , -\mathrm{maxd}\ , \mathrm{n} - 1)$

$$Bs := x \mapsto sum\_helper(helper(x), -maxd, n-1) \tag{2.2}$$

plot([*Bs*, *f*], 0 ..1, *color* = [*red*, *blue*])



Procedure

```
B_spline := proc(f)
 local helper, Bs
 if maxd = 2 then
   use_book_c(f);
else
   use_lesson_c(f);
end if;
 helper := x → j → ci(j) * B[j, maxd](x);

 Bs := x → sum_helper(helper(x), −maxd, n − 1);
return Bs
 end:
```

Оценки результата

Some init

```
with(Student[NumericalAnalysis]) :
with(CurveFitting) :
check_error := proc(f, approx)
 local nh, nn, i, err;
  err := 0;
  nn := n·10;
  nh := 1/nn;
  for i from 0 to nn do
    err := max(err, abs(f(i·nh) − approx(i·nh)));
  end ;
  return evalf(err);
 end:
value2pair := f → (x → [x, f(x)]) :


Cubic_spline_maple := proc(f)
 local v, i;
v := Array(0 .. n);
for i from 0 to n do
   v[i] := value2pair(f) (t[i])
 end;

 return (c → ApproximateValue(CubicSpline(convert(v, list), independentvar = x), c )[1][2])
 end:
cubic_splines := f→ (Cubic_spline(f), Cubic_spline_maple(f) ) :
my_splines := f→ (Cubic_spline(f), B_spline(f) ) :
```

sin(12x) (проверка работоспособности)

$f := x \to \sin(12 \cdot x)$

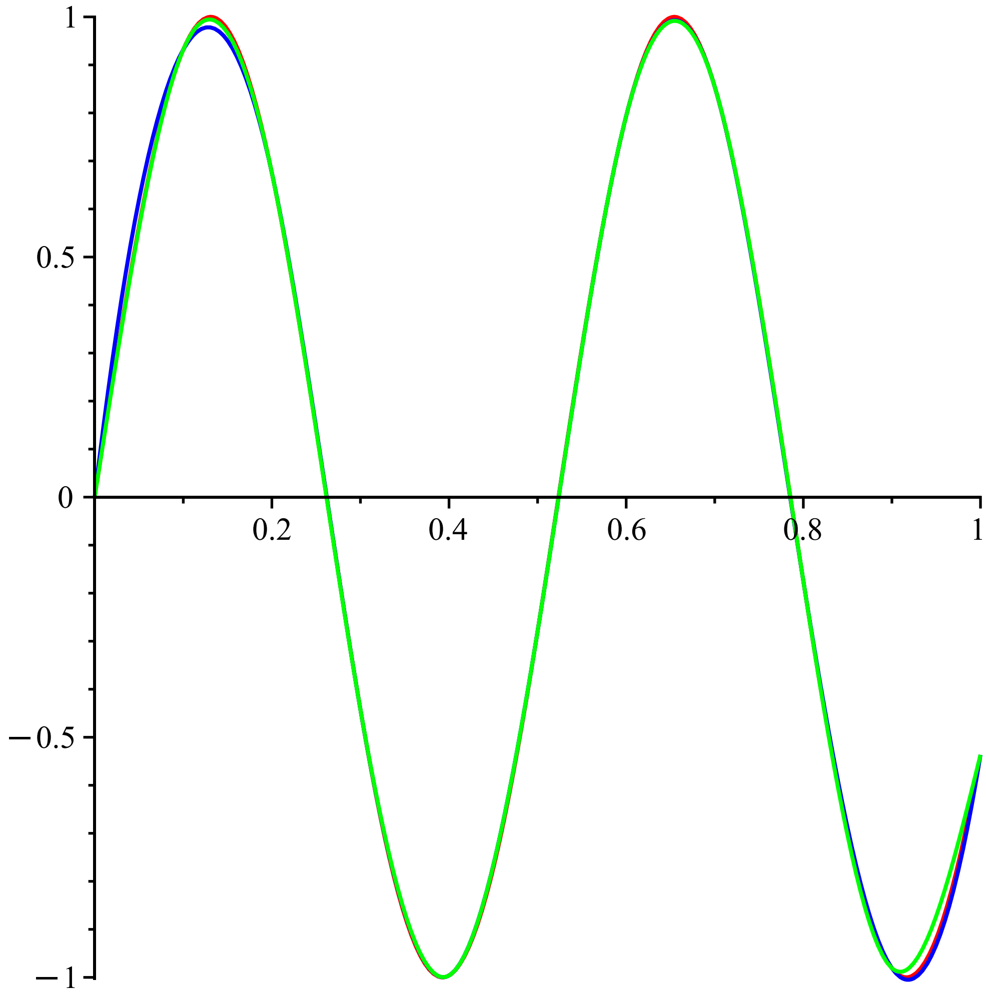
$$f := x \mapsto \sin(12 \cdot x) \tag{3.2.1}$$

Кубический

my, maple := *cubic_splines*(*f*)

$$my, maple := Cs, c \mapsto \left(ApproximateValue(CubicSpline(convert(v, list), independentvar = x), c)_1\right)_2 \tag{3.2.1.1}$$

plot([*f*, *my*, *maple*,], 0 ..1, *color* = [*red*, *blue*, *green*])



$check\_error(f, my)$

$$0.05888591235 \quad \textbf{(3.2.1.2)}$$

$check\_error(f, maple)$

$$0.0492907485434804 \quad \textbf{(3.2.1.3)}$$

Кубический сплайн из мапла справился чуть лучше, но считаю написаную реализацию кубического сплайна корректной.

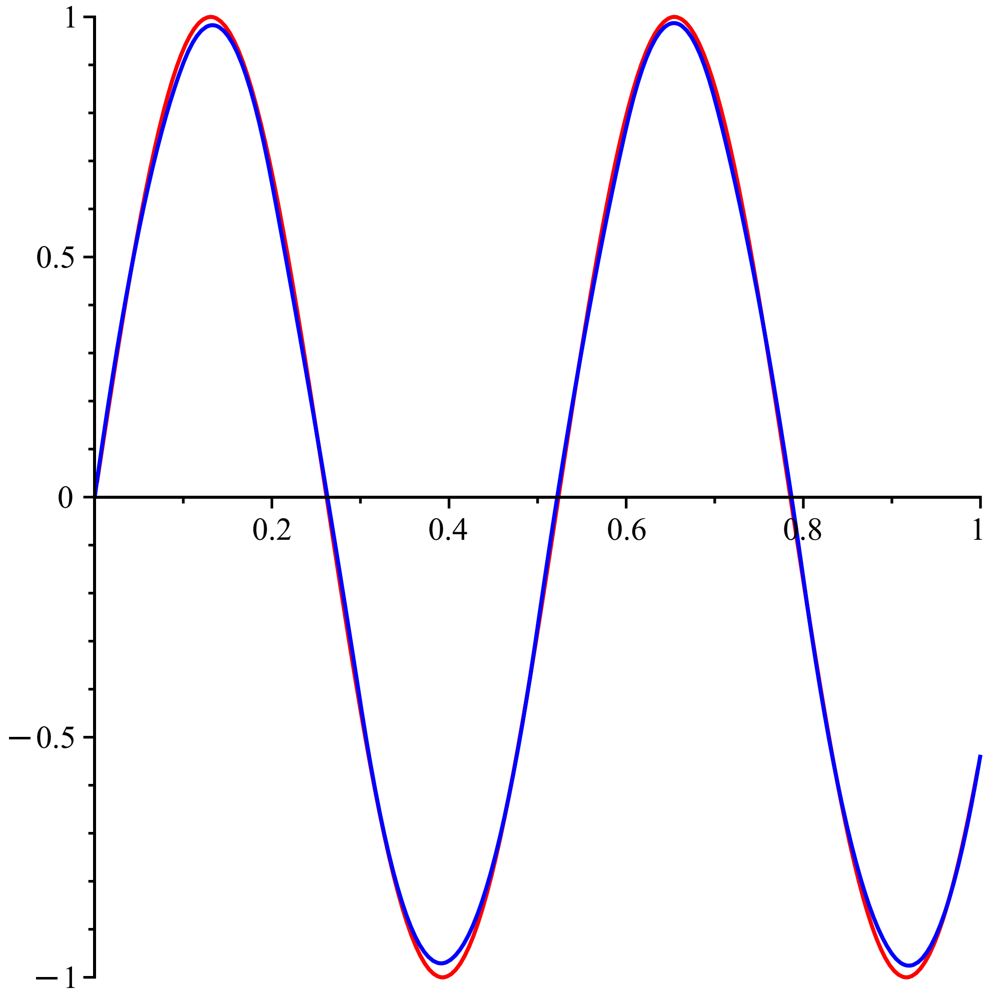
B spline

Сравнения с B-сплайном maple не будет, так как у меня не получилось завести мапловский.

$my := B\_spline(f)$

$$my, maple := Cs, c \mapsto \left(ApproximateValue(CubicSpline(convert(v, list), independentvar = x), c)_1\right)_2 \quad \textbf{(3.2.2.1)}$$

$plot([f, my], 0..1, color = [red, blue])$



$check\_error(f, my)$

$$0.0306501803 \tag{3.2.2.2}$$

Ошибка у B-сплайна ещё меньше чем у кубического, так что считаю его реализацию тоже можно считать упешной.

sin(1/x) (быстрая осциляция)

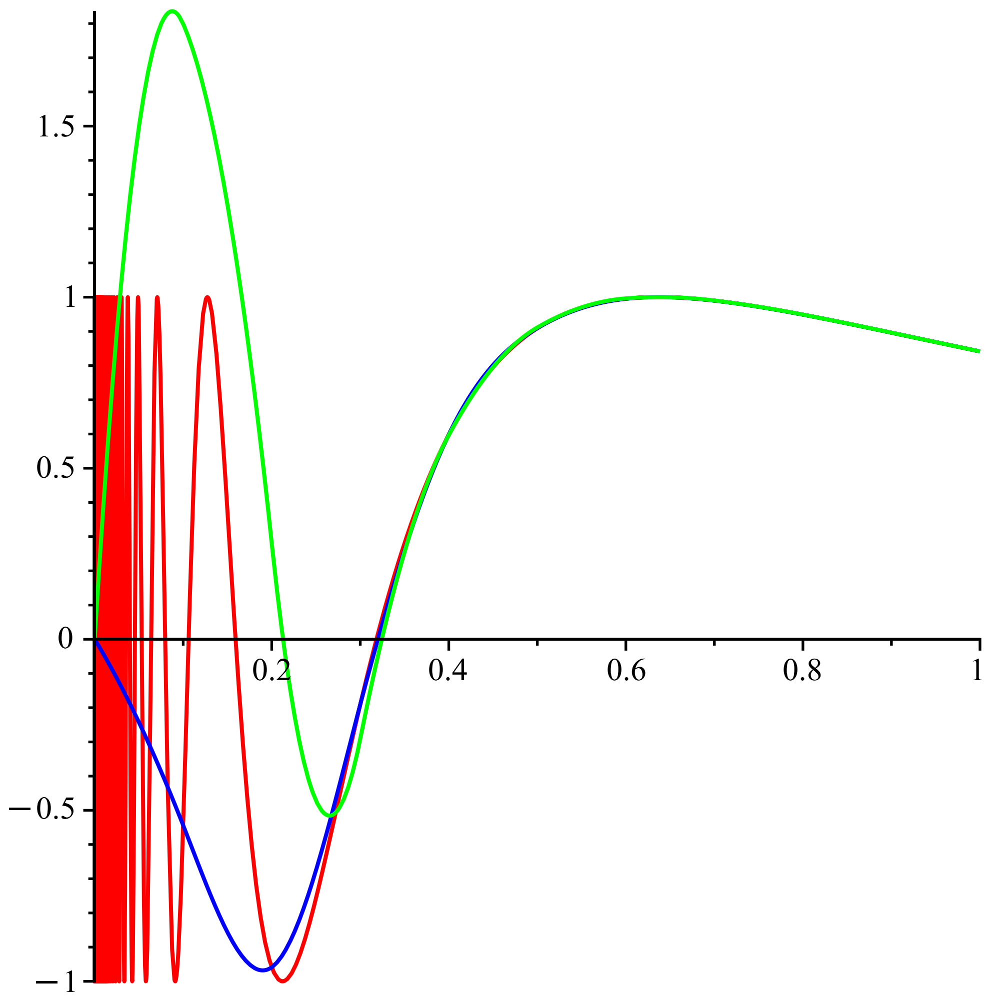
$f := x \rightarrow \textbf{if } x \neq 0 \textbf{ then } \sin\left(\frac{1}{x}\right) \textbf{ else } 0 \textbf{ end}$

$$f := x \mapsto \textbf{if } x \neq 0 \textbf{ then } \sin\left(\frac{1}{x}\right) \textbf{ else } 0 \textbf{ end if} \tag{3.3.1}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \tag{3.3.2}$$

$plot([f, cs, bs], 0..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$1.728925295 \tag{3.3.3}$$

$check\_error(f, bs)$

$$2.828014078 \tag{3.3.4}$$

Как и ожидалось сплайны плохо справляются с быстро осциллирующими функциями, как минимум из-за того что между отдельными значениями сетки, значение функции и её производных может несколько раз кардинально поменяться.

$\boldsymbol{abs(sin(x \cdot \pi \cdot 2))}$

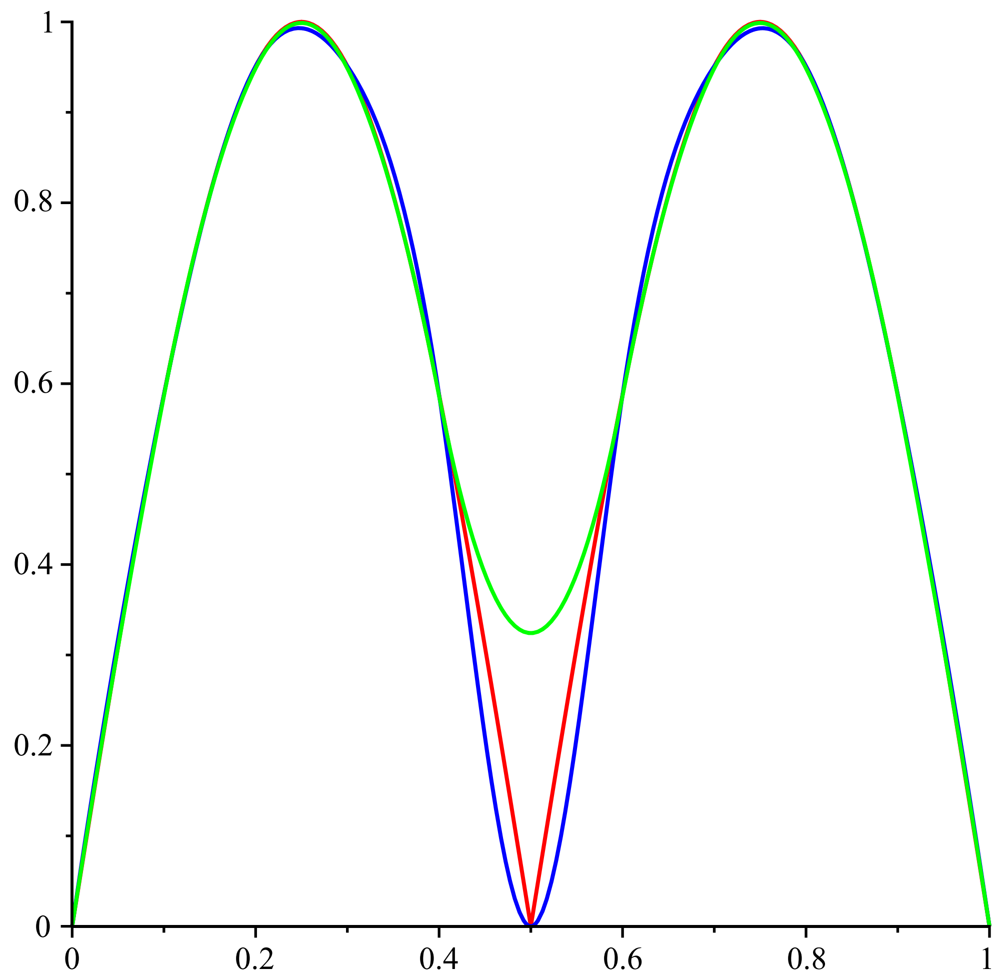
$f := x \rightarrow \mathrm{abs}(\sin(x \cdot \pi \cdot 2))$

$$f := x \mapsto |\sin(2 \cdot x \cdot \pi)| \tag{3.4.1}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \tag{3.4.2}$$

$plot([f, cs, bs], 0 ..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$0.2121481435 \tag{3.4.3}$$

$check\_error(f, bs)$

$$0.1473523828 \tag{3.4.4}$$

Плохо приближаются негладкие функции. На удивление B-spline справился хуже, хотя ожидалось что вызовет большую проблему для кубического

$\boldsymbol{abs(sin(1 + x \cdot \pi \cdot 2))}$

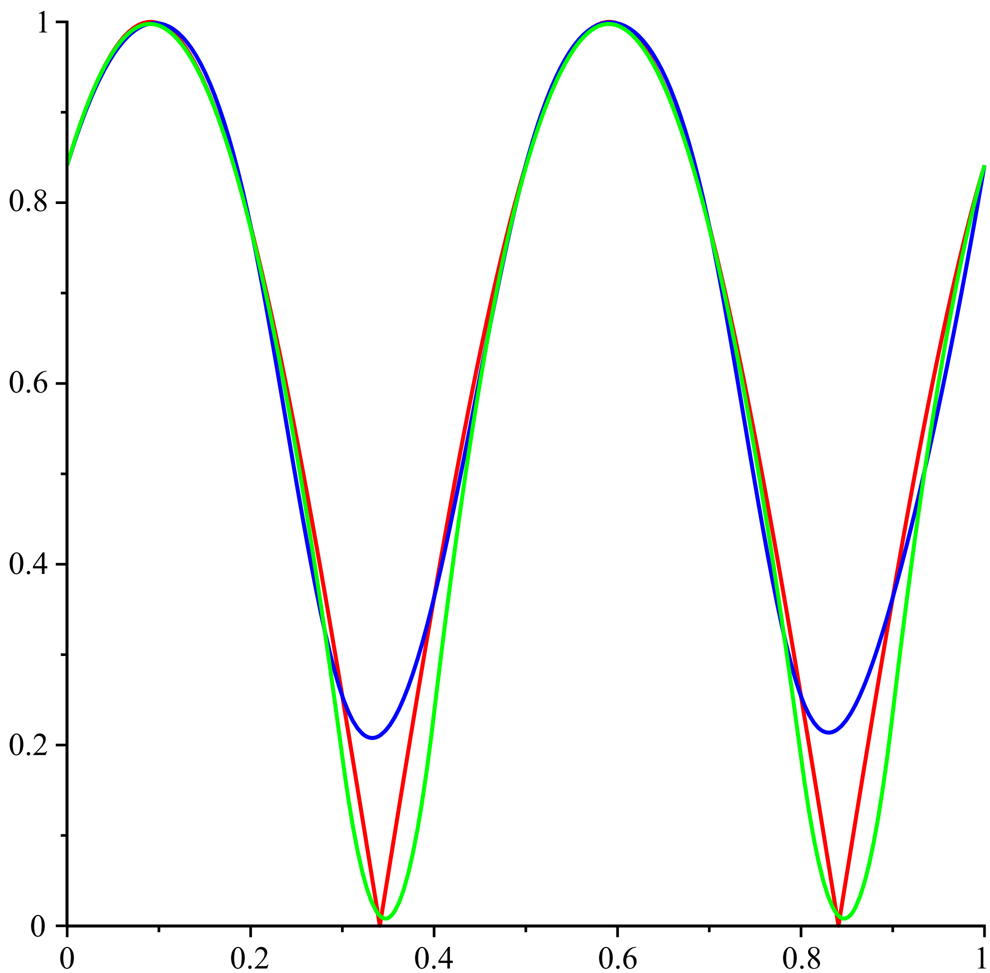
$f := x \to \operatorname{abs}(\sin(1 + x \cdot \pi \cdot 2))$

$$f := x \mapsto |\sin(1 + 2 \cdot x \cdot \pi)| \tag{3.5.1}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \tag{3.5.2}$$

$plot([f, cs, bs], 0 ..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$0.2121481435 \quad \textbf{(3.5.3)}$$

$check\_error(f, bs)$

$$0.1473523828 \quad \textbf{(3.5.4)}$$

Хотя если убрать точки, в которых график резко изменяется, из значений сетки (засчёт сдвига графика). То B-сплайны показывают себя лучше

e^{100x}

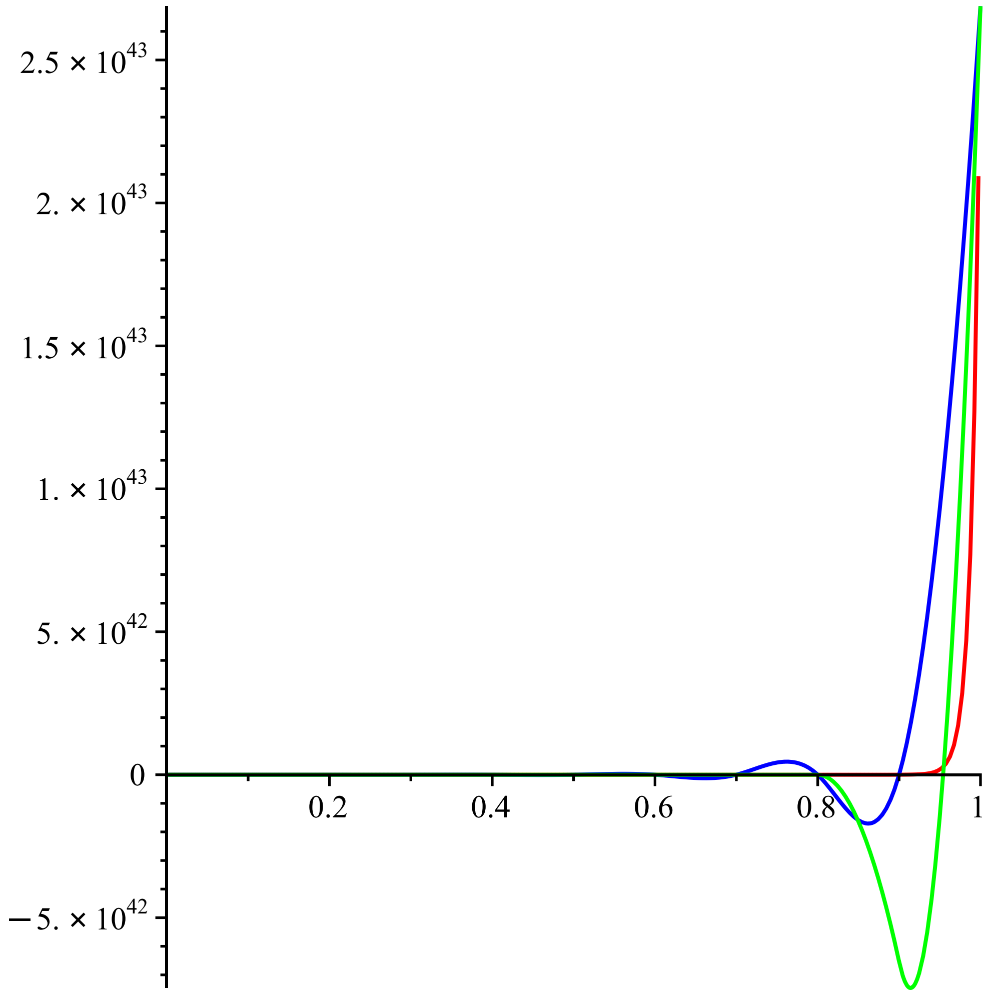
$f := x \to e^{100\,x}$

$$f := x \mapsto e^{100 \cdot x} \quad \textbf{(3.6.1)}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \quad \textbf{(3.6.2)}$$

$plot([f, cs, bs], 0 ..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$1.524903178 \times 10^{43} \tag{3.6.3}$$

$check\_error(f, bs)$

$$9.46518665 \times 10^{42} \tag{3.6.4}$$

Плохо приближаются быстро растующие функции. Забавный результат: целевая функция всюду положительна, но её интерполяция принимает отрицательные значения. Считаю, что на этой функции как и функции с быстрой осцилляцией сравнивать значения некоректно

$\boldsymbol{x^{\frac{1}{3}}}$

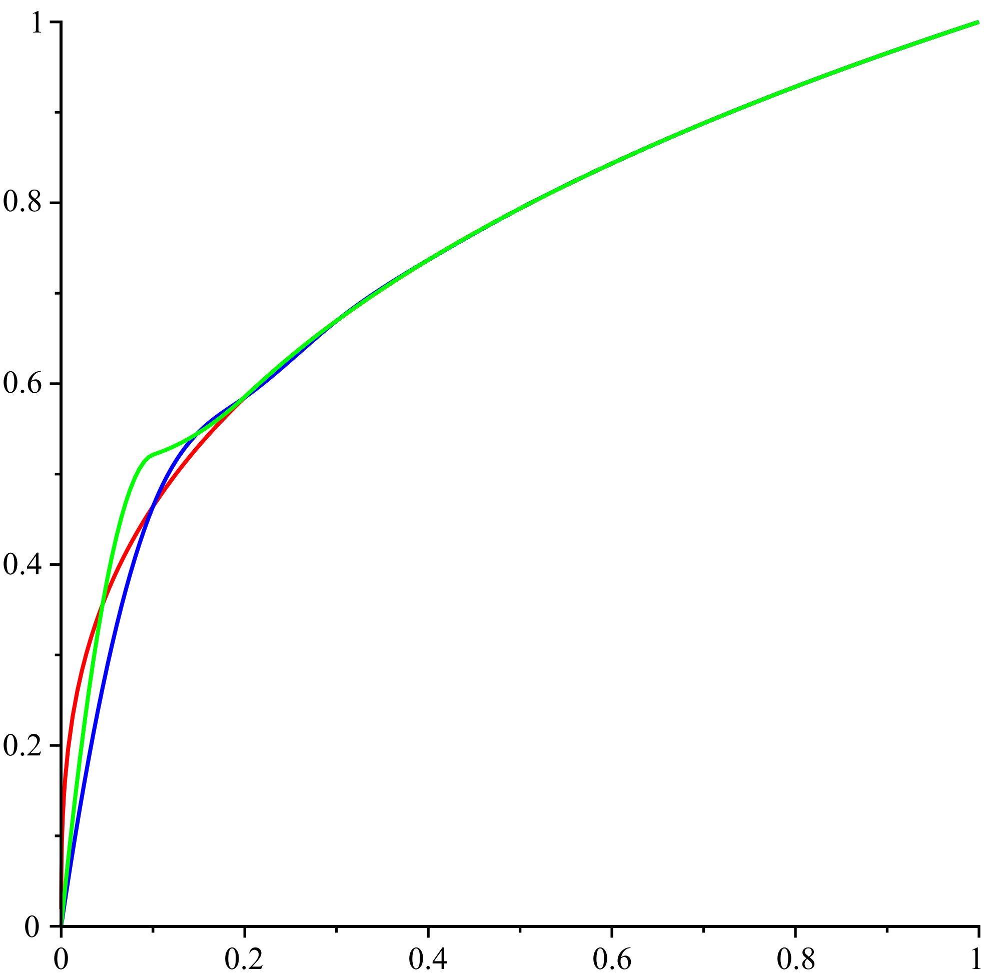
$f := x \rightarrow x^{\frac{1}{3}}$

$$f := x \mapsto x^{1/3} \tag{3.7.1}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \tag{3.7.2}$$

$plot([f, cs, bs], 0..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$0.1490415877 \tag{3.7.3}$$

$check\_error(f, bs)$

$$0.1193785224 \tag{3.7.4}$$

Как указано здесь https://www.proven-reserves.com/CubicSplines.php. "Резковертикальные" функции плохо приближаются кубическими сплайнами, оказывается это также распространяется и на B-сплайны. Но B-сплайн показал всё-таки более хороший результат (хотя по графику кажется наоборот)

Большая степень

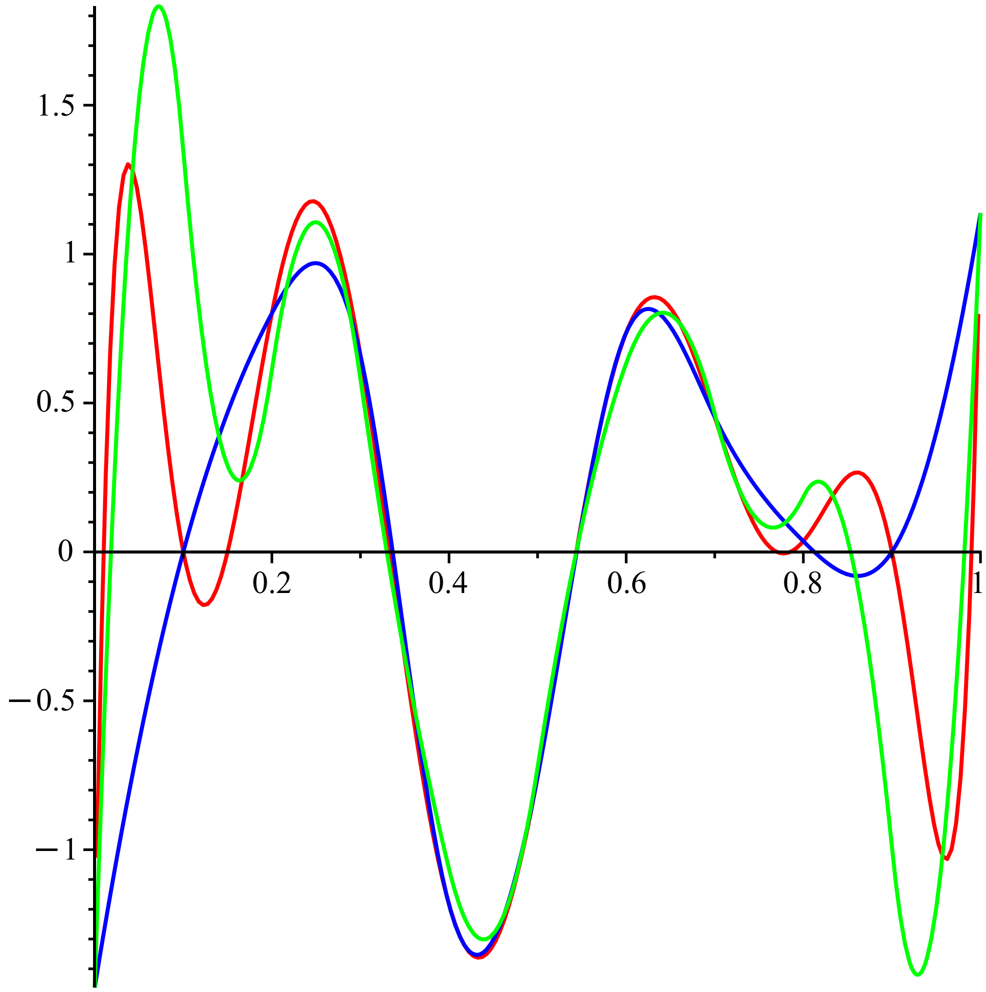
$$f := x \rightarrow 100000 \cdot \left((x - 0.1) \cdot \left(x - \frac{\pi}{4}\right) \cdot \left(x - \frac{e}{5}\right) \cdot (x - 0.77) \cdot (x - 0.333) \cdot (x - 0.01) \cdot (x - 0.99) \cdot (x - 0.15) \cdot (x - 0.9) \right)$$

$$f := x \mapsto 100000 \cdot (x - 0.1) \cdot \left(x - \frac{\pi}{4}\right) \cdot \left(x - \frac{e}{5}\right) \cdot (x - 0.77) \cdot (x - 0.333) \cdot (x - 0.01) \cdot (x - 0.99) \cdot (x - 0.15) \cdot (x - 0.9) \tag{3.8.1}$$

$cs, bs := my\_splines(f)$

$$cs, bs := Cs, Bs \tag{3.8.2}$$

$plot([f, cs, bs], 0 ..1, color = [red, blue, green])$



$check\_error(f, cs)$

$$2.166482552 \tag{3.8.3}$$

$check\_error(f, bs)$

$$1.451734073 \tag{3.8.4}$$

Из графика видно, что кроме того что b-сплайны более точны на полиномах большой степени, они также лучше сохраняют форму исходной функции чем кубические сплайны.