```
> restart
> with(plots) : with(LinearAlgebra) : with(geometry) :  with(MTM) :
>
> a := 0; b := 1 # Границы сетки
```

$$a := 0$$

$$b := 1 \tag{1}$$

```
> h1 := 1/10 # Шаг сетки
```

$$h1 := \frac{1}{10} \tag{2}$$

```
> X1 := [seq(a .. b, h1) ] # Сетка
```

$$X1 := \left[0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1\right] \tag{3}$$

CubicSpline

```
> Cubic_spline := proc(x, f, X, h)
local hi, hlp1, s1, k, hlp2, s2, hlp3, s3, M, yf, Y, Ck, Ak, i, helper, Dk, Fk, Bk, S,
    get_i, m;

m := nops(X) ; # Количество узлов
hi := h;  # Т.к. сетка равновемерная hi == hj ;i != j

# Инициализация трехдиагональной матрицы
hlp1 := i→ if ((i = 1) ) then 0 else hi end if; s1 := [seq(hlp1(k), k = 1 ..m − 1)] ;
    # i+1 −ые элементы
hlp2 := i→ if ((i = 1)or (i = m)) then 1 else 2·(hi + hi)end if; s2 :=
    [seq(hlp2(k), k = 1 ..m)] ;  # i −ые элементы
hlp3 := i→ if ((i = m − 1) ) then 0 else hi end if; s3 := [seq(hlp3(k), k = 1 ..m
    − 1)] ; # i-1 −ые элементы
M := 1/6 · BandMatrix([s3, s2, s1]) ;

# Инициализация вектора с результатами
yf := a → (f(X[a + 1])  − f(X[a]) )/hi  −  (f(X[a])  − f(X[a − 1]))/hi ;
Y := Vector([0, seq(yf(k) , k = 2 ..m − 1), 0] ) ;

# Вычисление искомых коэффициентов через матричное
    представление (M·Ck = Y)
Ck := [ op(convert(LinearSolve(M, Y) , list)) ] ;

# Вычисления коэффициентов A
Ak := map(f, [seq(X[i], i = 2 ..m) ]) ;
```

```
# Вычисления коэффициентов D
helper := i→ (Ck[i] − Ck[i − 1])/hi ;
Dk := map(helper, [seq(i, i = 2 ..m)]) ;

# Вычисления коэффициентов B
Fk := map(f, X) ;
helper := i→ (Fk[i] − Fk[i − 1])/hi + Ck[i]·(hi/3) + Ck[i − 1]·(hi/6) ;
Bk := map(helper, [seq(i, i = 2 ..m)]) ;

# Запись формулы кубического сплайна в явном виде
S := (i, x) → Ak[i] + Bk[i]·(x − X[i + 1]) + Ck[i + 1]/2·(x − X[i + 1])^2 + Dk[i]/6
    ·(x − X[i + 1])^3;
get_i := x → floor(x/hi) + 1; # Получение индекса коэффициентов
S(get_i(x), x);
end proc:
>
```

B-Spline

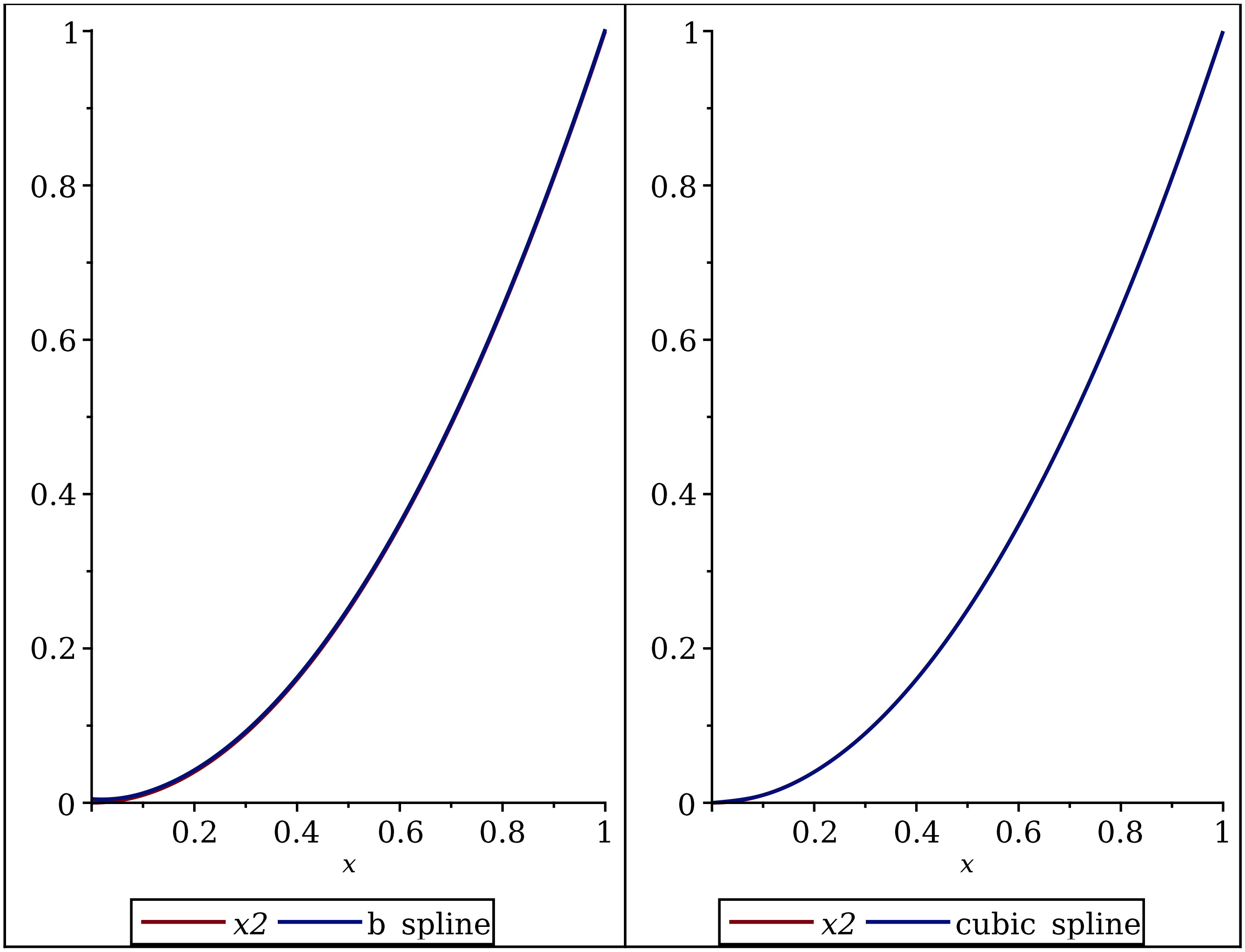
```
> B_spline := proc(x, f, X, h)
local dm, X_ext, B_i0, B_ij, B_i1_, B_i2_, B_i2, fc, Ck, m;
with(Statistics) ;

m := nops(X) ; # Количество узлов
dm := 2; # Порядок B-сплайна

# Создание расширенной вправо и влево сетки
X_ext := [seq(a − h·i, i = 1 ..dm) , op(X), seq(b + h·i, i = 1 ..dm + 3) ] ;

# Задание B-сплайнов до второго порядка
B_i0 := (x, i, X, d)→piecewise(X[i] <  x ≤ X[i + 1], 1, 0) ;
B_ij := (x, i, X, d, b)→ (x − X[i])/(X[i + d] − X[i]) ·b(x, i, X, d − 1) + (X[i + d + 1] − x)/(X[i + d + 1] − X[i + 1])
    ·b(x, i + 1, X, d − 1) ;

B_i1_ := (x, i, X, d)→ B_ij(x, i, X, d, B_i0) ;
B_i2_ := (x, i, X, d) → B_ij(x, i, X, d, B_i1_) ;
B_i2 := (x, i)→ B_i2_(x, i, X_ext, 2) ;

# Вычисление коэффициент
fc := i→ Mean( map(k→ X_ext[k], [seq(i + j, j = 1 ..dm)] ) ) ;
Ck := [op(map(i→f(fc(i)) , [seq(i, i = 1 ..m + 1)])) ] ;
```

```
# Описание формулы B-сплайна через суммирование
add(Ck[i]·B_i2(x, i) , i = 1 ..m + 1) ;
end proc:
>
> cubic_spline := (x, f)→ Cubic_spline(x, f, X1, h1) :
> b_spline := (x, f)→ B_spline(x, f, X1, h1) :
>
> # Функции для вычисления ошибки на сетке в 10 раз меньше исходной
> get_error := (f, f_res) → max(map(x→evalf(abs(f(x, f_res) − f_res(x))),
      [seq(a ..(b − 0.01), h1 · 1/10 )])) :
> get_cubic_err := f → get_error(cubic_spline, f) :
> get_b_err := f → get_error(b_spline, f) :
> get_i := x → floor(x·10) + 1 :; map(x→evalf(get_i(x)), [seq(a ..(b − 0.01), h1
      · 1/10 )]) :
> # Функция для создания графиков функций
> create_plot := f → Array([
    plot([f(x), b_spline(x, f)], x = a ..b, legend = [f , "b_spline"]),
    plot([f(x), cubic_spline(x, f)], x = a ..b, legend = [f, "cubic_spline"])
  ]):
>
```

(,
)

```
>
> x2 := x→ x^2 ; display(create_plot(x2));
```

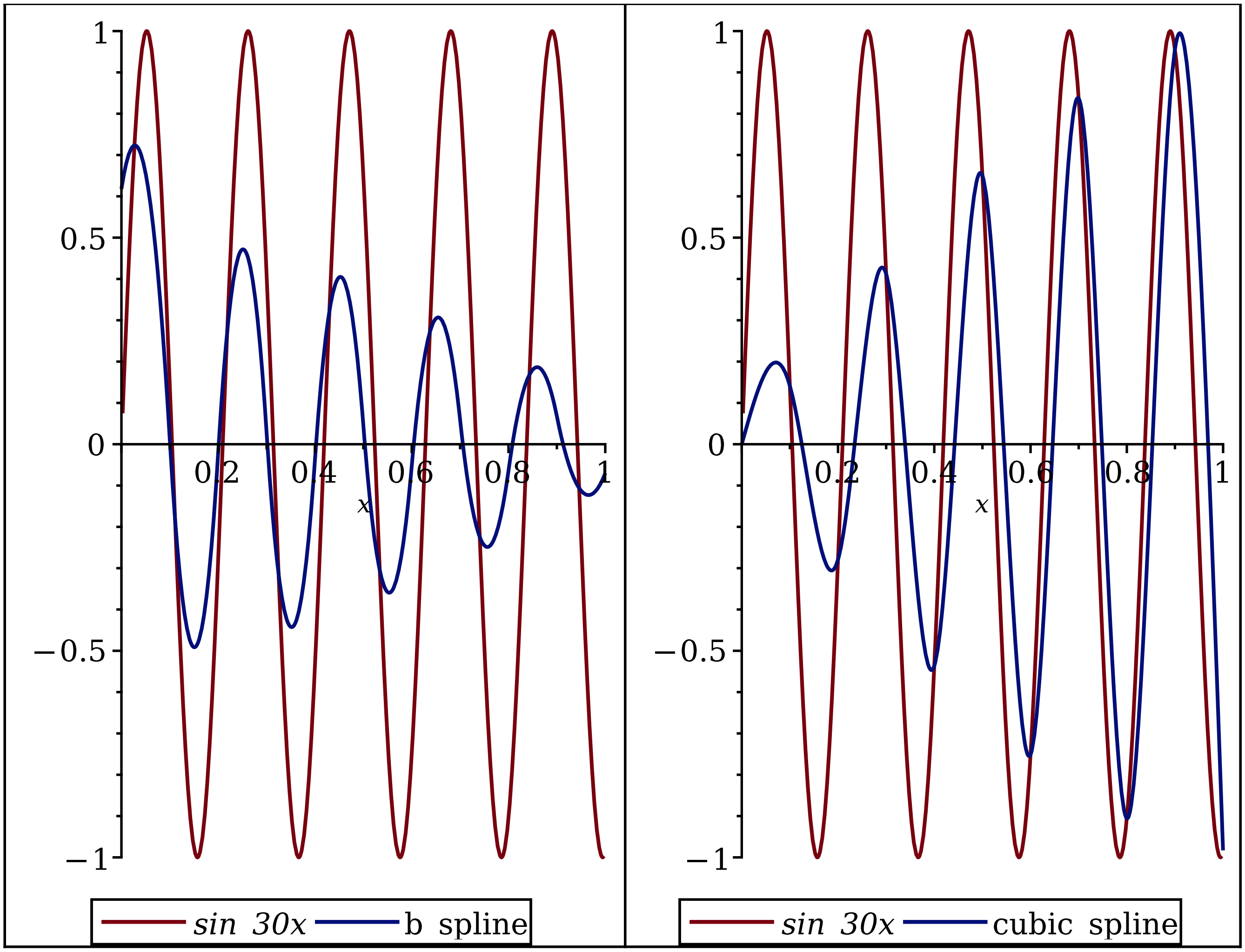
$$x2 := x \mapsto x^2$$



> # *Рассмотрим переодическую функцию с "высокой" частотой. Предлагаемая функция* $-\ sin(40 \cdot x)$

> $sin\_30x := x \rightarrow \sin(30 \cdot x)$; $display(create\_plot(sin\_30x))$

$$sin\_30x := x \mapsto \sin(30 \cdot x)$$



> # *Обратим внимание, что ни одна из реализаций сплайнов не аппроксимируют эту функцию достаточно хорошо, подтверждением являются абсолютные ошибки B_сплайна и кубического сплайна (соответственно)*

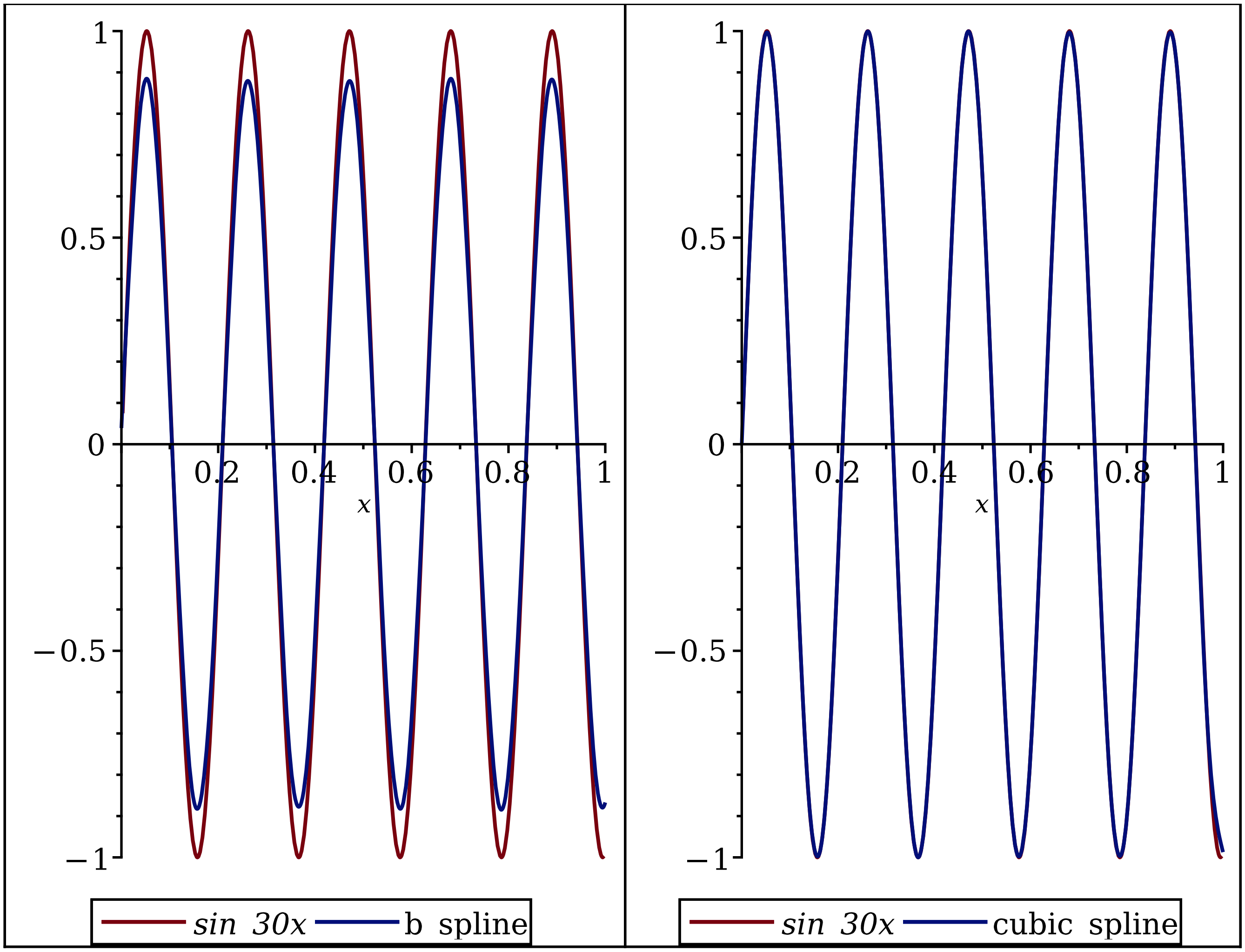
> $[get\_b\_err(sin\_30x), get\_cubic\_err(sin\_30x)]$

$$[0.893276201915059, 0.8226728499] \quad \textbf{(4)}$$

> # *Однако, если уменьшить сетку в 3 раза, то проблема пропадет и аппроксимация будет приемлимой.*

> $h2 := \frac{1}{30} : X2 := [\mathrm{seq}(\mathrm{a} .. \mathrm{b}, \mathrm{h2})]:$

> $display(Array([$
$plot([sin\_30x(x), B\_spline(x, sin\_30x, X2, h2)], x = a..b, legend = [sin\_30x,$ "b_spline"]),
$plot([sin\_30x(x), Cubic\_spline(x, sin\_30x, X2, h2)], x = a..b, legend$ $= [sin\_30x,$ "cubic_spline"])
$]))$



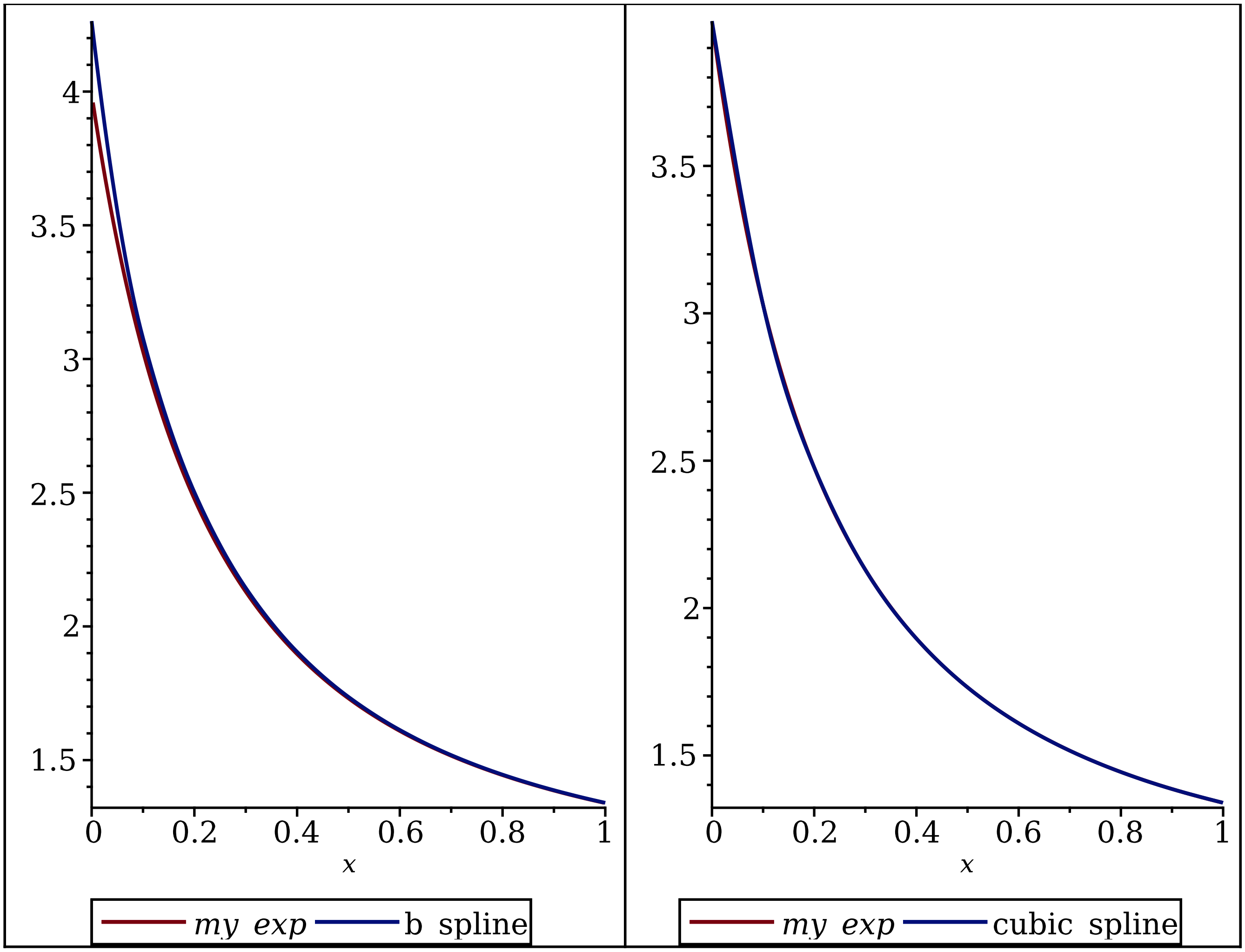
> # *Вывод: успех аппроксимации высокочастотных, переодических функций зависит от выбранного шага сетки.*

> # *Еще рассмотрим exp* $\frac{1}{(x+0.4)^2}$ *. Эта функция показывает, что размер ошибки зависит от величины функции, что довольно таки очевидно . Однако ошибка у B_сплайна около границ больше, чем у кубического сплайна.*

> # *Так происходит из−за малого количества узлов сетки (при быстром изменении самой функции).*
Стоит отметить, что B_сплайны могут вести себя заметно хуже кубических сплайнов около границ (такова цена динамического построения).

> $my\_exp := x \rightarrow \exp\left(\frac{1}{(x+0.85)^2}\right); display(create\_plot(my\_exp))$

$$my\_exp := x \mapsto \mathrm{e}^{\frac{1}{(x+0.85)^2}}$$




```
> [get_b_err(my_exp), get_cubic_err(my_exp)]
    # Ошибка б-сплайна и кубического сплайна.
```

[0.272328770577249, 0.0391986047545441] **(5)**