> $with(CurveFitting);$

$$[ArrayInterpolation, BSpline, BSplineCurve, Interactive, LeastSquares, Lowess, PolynomialInterpolation, RationalInterpolation, Spline, ThieleInterpolation] \quad (1)$$

> $N := 10;$

$$N := 10 \quad (2)$$

> $k := 3$ # *additional knots*

$$k := 3 \quad (3)$$

> $d := 2$

$$d := 2 \quad (4)$$

> $X := Array\left(\left[seq\left(-\frac{k-i+1}{10^{10}}, i=1..k\right), seq\left(\frac{i}{N}, i=0..N\right), seq\left(1+\frac{i}{10^{10}}, i=1..k\right)\right]\right)$

$$X := \begin{bmatrix} -\frac{3}{10000000000} & -\frac{1}{5000000000} & -\frac{1}{10000000000} & 0 & \frac{1}{10} & \cdots \end{bmatrix} \quad (5)$$

> $B := \mathbf{proc}(i\_, d, t)$
local i;
$i := i\_$;
if $i < 1$ **then** $i := 1 + k$ **elif** $i > N + 1 + 2k$ **then** $i := N + k$; **end if**;
if $d = 0$ **then** $\begin{cases} 1 & X[i] \le t < X[i+1] \\ 0 & otherwise \end{cases}$ **else** $\frac{(t - X[i])}{X[i+d] - X[i]} \cdot B(i\_, d-1, t)$
$+ \frac{(X[i+d+1] - t)}{X[i+d+1] - X[i+1]} \cdot B(i\_ + 1, d-1, t)$ **end if**
end proc;

```
B := proc(i_, d, t)
    local i;
    i := i_;
    if i < 1 then i := 1 + k elif N + 1 + 2*k < i then i := N + k end if;
    if d = 0 then
        piecewise(X[i] <= t and t < X[i + 1], 1, 0)
    else
        (t - X[i])*B(i_, d - 1, t)/(X[i + d] - X[i]) + (X[i + d + 1] - t)*B(i_ + 1,
        d - 1, t)/(X[i + d + 1] - X[i + 1])
    end if
end proc                                                                  (6)
```

> $get\_c := \mathbf{proc}(i, f)$
local $c, eqs, j, r, Solved, x, get\_f$;
$get\_f := \mathbf{proc}(m, x)$
$add(b[j] \cdot B(j, d, op(m, x)), j = i-1..i+1)$
end proc;
$x := [0, 0, 0]$;
$x[1] := X[i+1]$;

```
x[2] := (X[i + 1] + X[i + 2])/2;
x[3] := X[i + 2];
eqs := [seq(f_approx[op(r, x)] = add(b[j]·B(j, d, op(r, x)), j = i − 1 ..i + 1), r = 1 ..3)];
eqs := [op(eqs), seq(f_approx[op(r, x)] = f(op(r, x)), r = 1 ..3)];
Solved := solve(eqs);
if i ≤ 1 then c := f(X[k + 1]); elif i ≥ N + 1 + 2 k then c := f(X[N + k]) else c :=
    eval(1/2 · (−get_f(1, x) + 4·get_f(2, x) −get_f(3, x)), Solved) end if;
c
end proc;
```

get_c := **proc**(i, f) **(7)**

```
    local c, eqs, j, r, Solved, x, get_f;
    get_f := proc(m, x) add(b[j]*B(j, d, op(m, x)), j = i − 1 ..i + 1) end proc;
    x := [0, 0, 0];
    x[1] := X[i + 1];
    x[2] := 1/2*X[i + 1] + 1/2*X[i + 2];
    x[3] := X[i + 2];
    eqs := [seq(f_approx[op(r, x)] = add(b[j]*B(j, d, op(r, x)), j = i − 1 ..i + 1), r = 1 ..3)];
    eqs := [op(eqs), seq(f_approx[op(r, x)] = f(op(r, x)), r = 1 ..3)];
    Solved := solve(eqs);
    if i <= 1 then
        c := f(X[1 + k])
    elif N + 1 + 2*k <= i then
        c := f(X[N + k])
    else
        c := eval( − 1/2*get_f(1, x) + 2*get_f(2, x) − 1/2*get_f(3, x), Solved)
    end if;
    c
end proc
```

```
> B_spline := proc(i, t, f)
  B(i, d, t) · get_c(i, f);
  end proc;
```

B_spline := **proc**(i, t, f) B(i, d, t)*get_c(i, f) **end proc** **(8)**

```
> P := proc(t, f)
  local i;
  eval(add(B_spline(i, t, f), i = 2 ..N + 3));
  end proc;
```

P := **proc**(t, f) **local** i; eval(add(B_spline(i, t, f), i = 2 ..N + 3)) **end proc** **(9)**

```
> plotF := proc(f)
  local XY, c, i;
  XY := [seq([X[i], f(X[i])], i = 1 .. N + 2 k)];
```

```
c := BSplineCurve(XY, x, order = 3);
plot({c, P(x, f), f(x)}, x = 0 ..1 , color = [red, green, blue]);
end proc;
```

$$plotF := \mathbf{proc}(f) \tag{10}$$

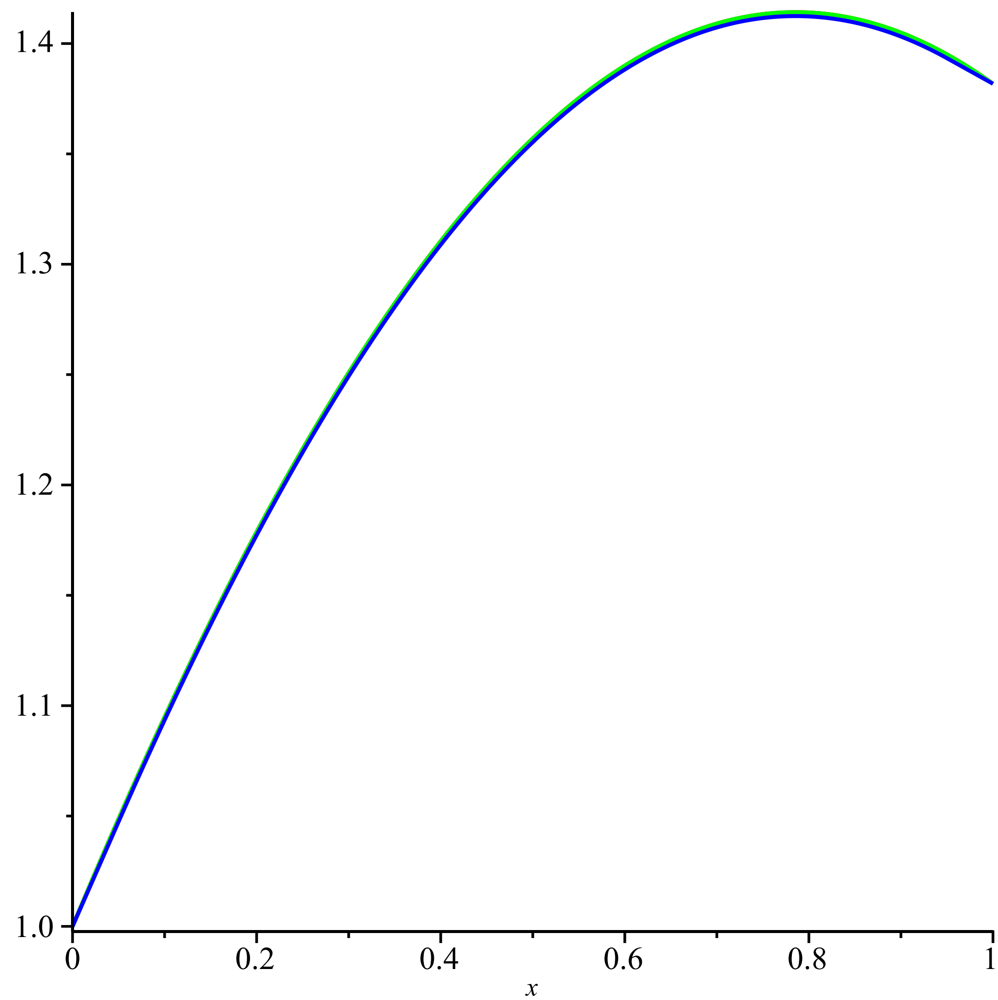
```
    local XY, c, i;
    XY := [seq([X[i], f(X[i])], i = 1 ..N + 2 * k)];
    c := CurveFitting:-BSplineCurve(XY, x, order = 3);
    plot({P(x, f), f(x), c}, x = 0 ..1, color = [red, green, blue])
end proc
```

Красное — реализация B-сплайна с использованием коэффициентов из известной работы, зеленое — аппроксимируемая функция, синее — стандартная реализация при d=3.

> $f := x \to \sin(x) + \cos(x)$

$$f := x \mapsto \sin(x) + \cos(x) \tag{11}$$

> $plotF(f);$

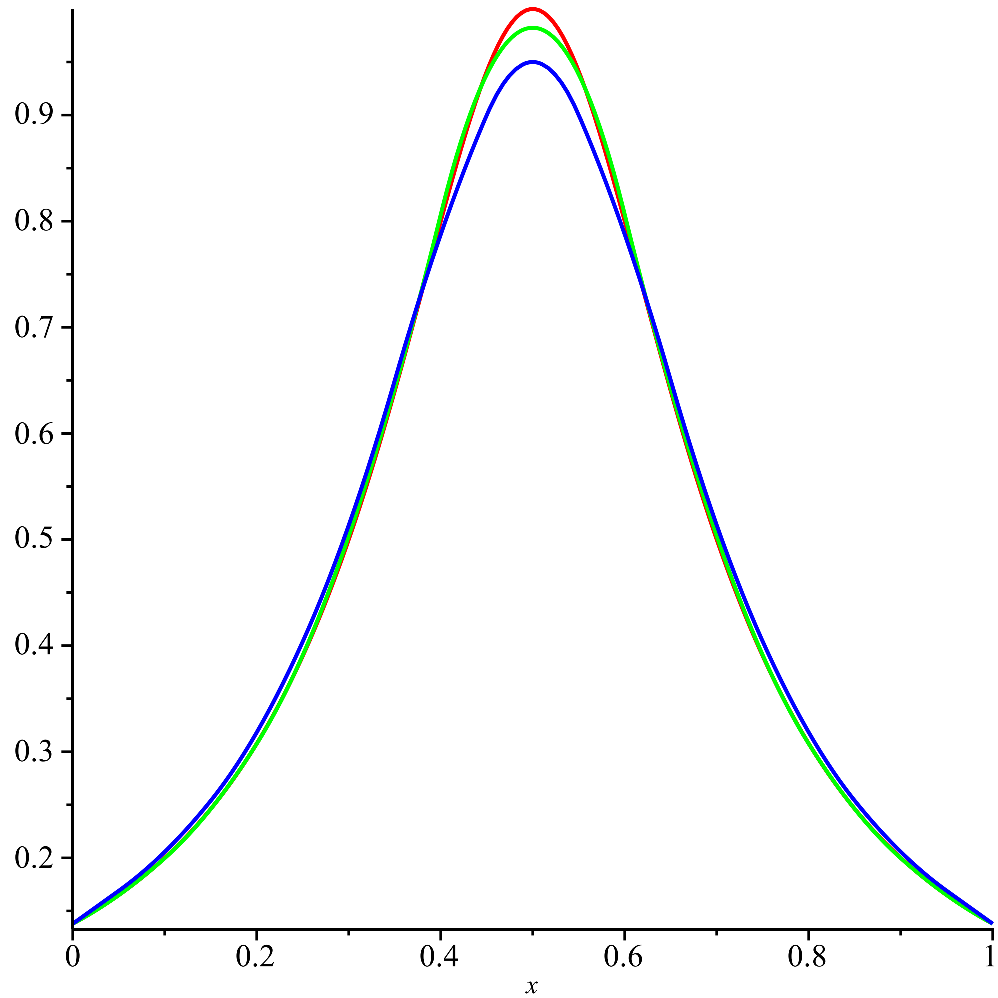


> $f := x \to \dfrac{1}{1 + 25\left(x - \frac{1}{2}\right)^2}$

(12)

$$f := x \mapsto \frac{1}{1 + 25 \cdot \left(x - \frac{1}{2}\right)^2} \tag{12}$$

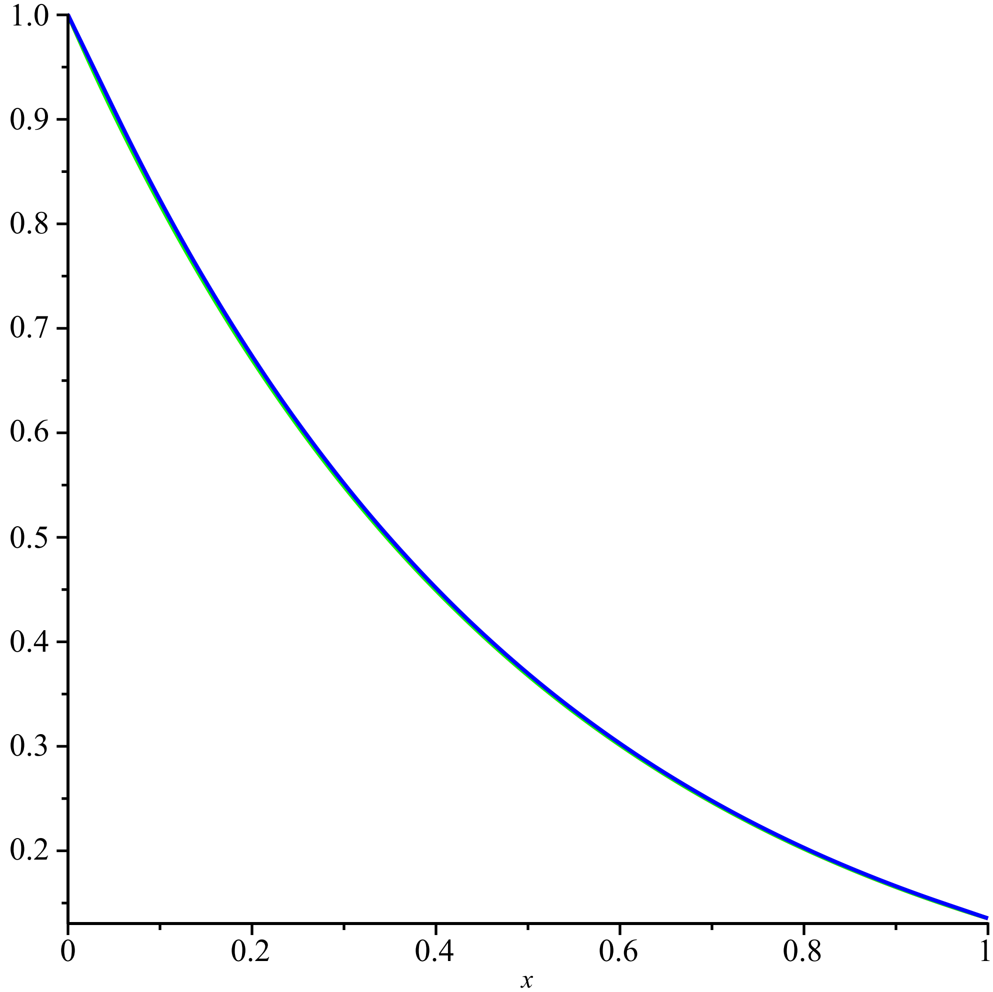
> *plotF*(f);
> *# Получили более точное приближение функции Рюнге!*



> $f := x \to \exp(-2\,x)$

$$f := x \mapsto e^{-2 \cdot x} \tag{13}$$

> *plotF*(f)



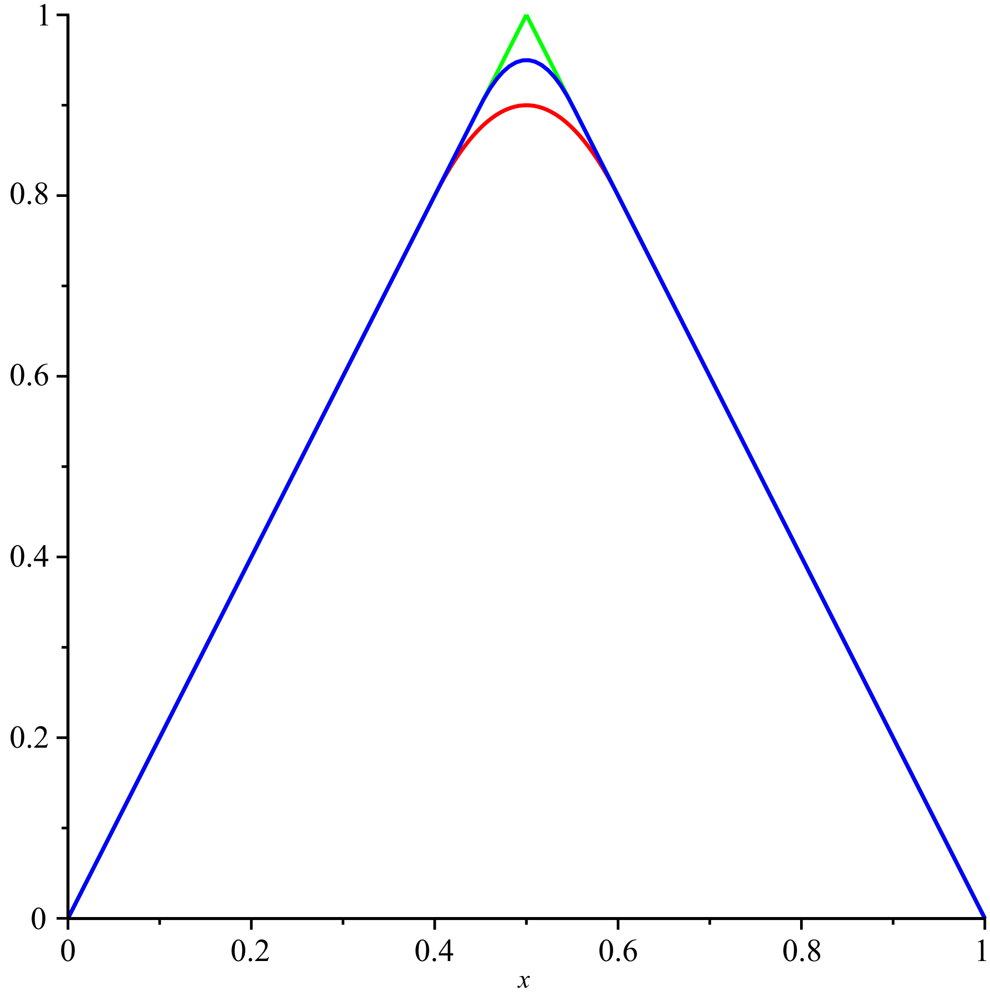
> $f := x \rightarrow \begin{cases} 2\,x & x \leq 0.5 \\ -2\,x + 2 & x \geq 0.5 \end{cases}$

 # Проверим свойство формосохранения. Теоретическое ожидание — более точная форма с использованием B-сплайнов.

$$f := x \mapsto \begin{cases} 2 \cdot x & x \leq 0.5 \\ -2 \cdot x + 2 & 0.5 \leq x \end{cases} \tag{14}$$

> $plotF(f)$

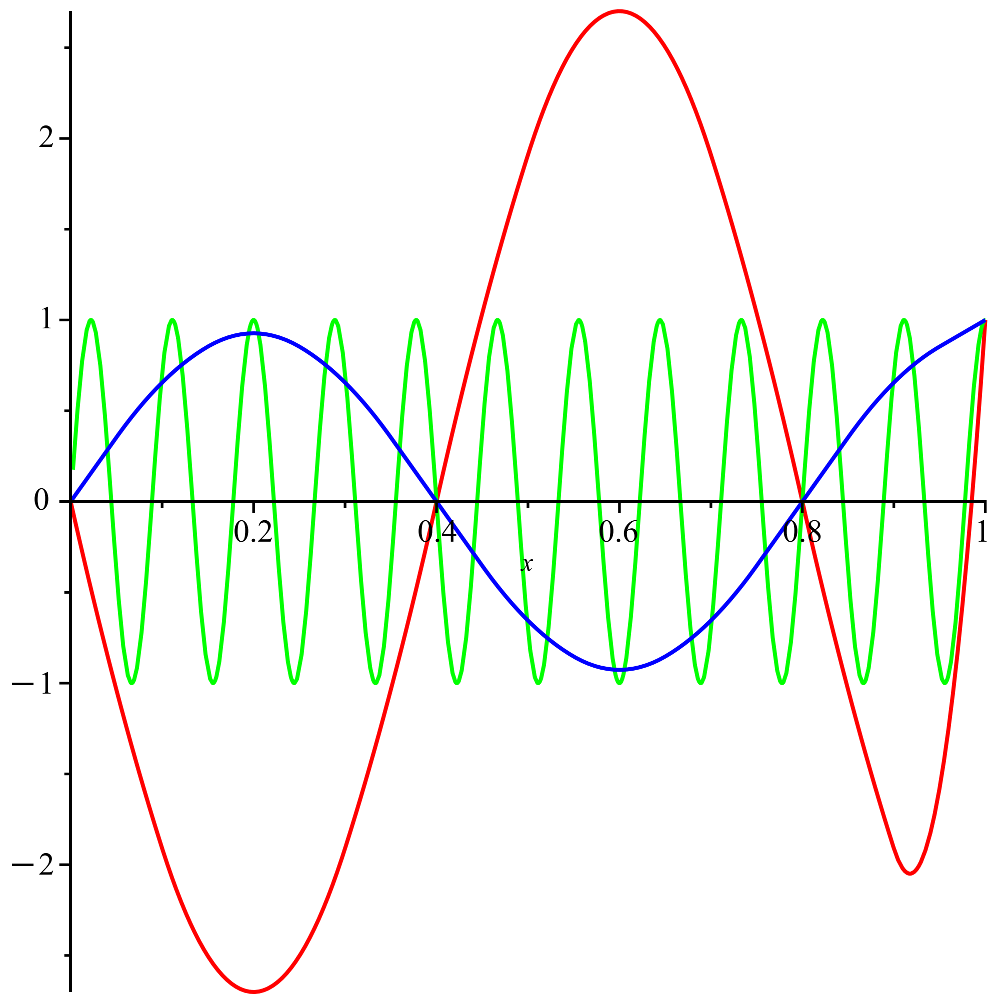
 # Хоть и B-сплайны на этом примере также обладают проблемой с потерей формы, однако, справедливости ради, при уточнении большим количеством точек с использованием B-сплайнов получится более точная форма, что подтверждает теоретические ожидания.



> $f := x \rightarrow \sin\left(\frac{\mathrm{Pi}}{2} \cdot \left(N + \frac{70}{2}\right) \cdot x\right)$

$$f := x \mapsto \sin\left(\frac{\pi \cdot (N + 35) \cdot x}{2}\right) \tag{15}$$

> *plotF*(*f*)
 # Здесь все совсем плохо, кубические сплайны справились лучше.



2
1
0
−1
−2
0.2
0.4
0.6
0.8
1
x