```
> restart;
  interface(warnlevel = 0);
                                   3                                   (1)
> with(LinearAlgebra) :

  spline3 := proc(f)
  segment := 0 ..1;
  n := 10;
  h := 0.1;
  xs := seq(i, i = segment, h);
  ys := seq(f(i), i = segment, h);

  A_init := (i, j)→if (i = 1 and j = 1) then 1 elif (i = 1 and j = 2) then 0 elif (i = n + 1 and j = n
      + 1) then 1 elif (i = n + 1 and j = n) then 0 elif i = j then 4·h elif abs(i − j) = 1 then h
      else 0; end if;
  A := Matrix(n + 1, A_init);
  B_init := j→if (j = 1 or j = n + 1) then 0 else 1/h ((ys[j + 1] − ys[j]) − (ys[j ]− ys[j
      − 1])); end if;
  B := 6· Vector(n + 1, B_init);
  c := LinearSolve(A, B);
  a := seq(ys[i], i = 2 ..n + 1);
  d := seq((c[i] − c[i − 1])/h, i = 2 ..n + 1);
  b := seq((ys[i] − ys[i − 1])/h + c[i]·h/3 + c[i − 1]·h/6, i = 2 .. n + 1);
  S := seq(a[i] + b[i]·(x − xs[i + 1]) + c[i + 1]/2·(x − xs[i + 1])^2 + d[i]/6·(x − xs[i
      + 1])^3, i = 1 ..n);
  return piecewise(0 ≤ x < 0.1, S[1], 0.1 ≤ x < 0.2, S[2], 0.2 ≤ x < 0.3, S[3], 0.3 ≤ x
      < 0.4, S[4], 0.4 ≤ x < 0.5, S[5], 0.5 ≤ x < 0.6, S[6], 0.6 ≤ x < 0.7, S[7], 0.7 ≤ x
      < 0.8, S[8], 0.8 ≤ x < 0.9, S[9], 0.9 ≤ x ≤ 1, S[10] );
  end proc:

  spline3_sub := proc(f, b)
  q1 := subs(x = b, spline3(f) (x));
  return eval(q1);
  end proc;

  splineB := proc(f)
  segment := 0 ..1;
  n := 12;
  h := 0.1;
  eps := 10^(−9);
  xs := [ −3·eps, −eps, seq(i, i = segment, h), 1 + eps, 1 + 3·eps];
  #ys := [f(0), f(0), seq(f(i), i = segment, h), f(1), f(1)];
```

```
lam := j→ piecewise(j = 1, f(xs[1]), 1 < j < n, 1/2 (−f(xs[j + 1])
    + 4 f((xs[j + 1] + xs[j + 2])/2) − f(xs[j + 2])), j = n, f(xs[n + 1]));
B0 := (i, x) → piecewise(xs[i] ≤ x < xs[i + 1], 1, 0);
B1 := (i, x) → (x − xs[i])/(xs[i + 1] − xs[i]) · B0(i, x) + (xs[i + 2] − x)/(xs[i + 2] − xs[i + 1]) · B0(i + 1, x);
B2 := (i, x) → (x − xs[i])/(xs[i + 2] − xs[i]) · B1(i, x) + (xs[i + 3] − x)/(xs[i + 3] − xs[i + 1]) · B1(i + 1, x);
return x→sum(lam(i) · B2(i, x), i = 1 ..n);
end proc:

splineB_sub := proc(f, b)
 q1 := subs(x = b, splineB(f) (x));
 return eval(q1);
end proc;

with(ArrayTools) :
check_deviations := proc(f)
  segment := seq(j, j = 0 .. 1, 0.1);
  deviations := Array([ ]);
   for i from 2 to 11 do
     xs := [seq(k, k = segment[i − 1] ..segment[i], 0.01)];
     diff := x → abs(f(x) − spline3_sub(f, x)) ;
     deviations := Append(deviations, max(map(diff, xs)));
  end do;
  return max(deviations);
end proc;

 with(ArrayTools) :
check_deviations_for_spline := proc(spline_sub, f)
  segment := seq(j, j = 0 .. 1, 0.1);
  deviations := Array([ ]);
   for i from 2 to 11 do
     xs := [seq(k, k = segment[i − 1] ..segment[i], 0.01)];
     diff := x → abs(f(x) − spline_sub(f, x)) ;
     deviations := Append(deviations, max(map(diff, xs)));
  end do;
  return deviation = max(deviations);
end proc;
spline3_sub := proc(f, b)
    local q1; q1 := subs(x = b, spline3(f) (x)); return eval(q1)
end proc
splineB_sub := proc(f, b)
    local q1; q1 := subs(x = b, splineB(f) (x)); return eval(q1)
end proc
```

```
check_deviations := proc(f)
    local segment, j, deviations, i, xs, k, diff;
    segment := seq(j, j = 0 ..1, 0.1);
    deviations := Array([ ]);
    for i from 2 to 11 do
        xs := [seq(k, k = segment[i − 1]..segment[i], 0.01)];
        diff := x→abs(f(x) − spline3_sub(f, x));
        deviations := ArrayTools:-Append(deviations, max(map(diff, xs)))
    end do;
    return max(deviations)
end proc
check_deviations_for_spline := proc(spline_sub, f)
    local segment, j, deviations, i, xs, k, diff;
    segment := seq(j, j = 0 ..1, 0.1);
    deviations := Array([ ]);
    for i from 2 to 11 do
        xs := [seq(k, k = segment[i − 1]..segment[i], 0.01)];
        diff := x→abs(f(x) − spline_sub(f, x));
        deviations := ArrayTools:-Append(deviations, max(map(diff, xs)))
    end do;
    return deviation = max(deviations)
end proc
```
(2)

> \# *Пример для фукнции типа Рунге :*

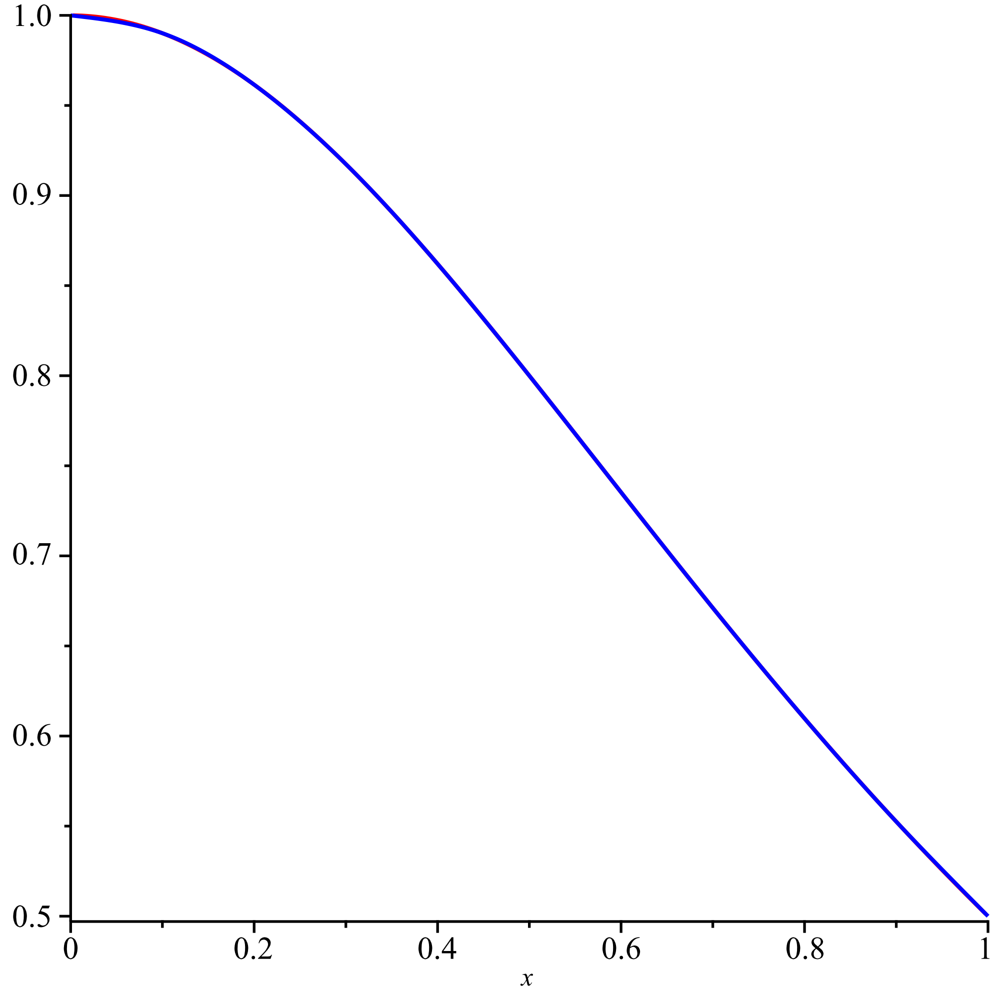
$$runge := x \rightarrow \frac{1}{1 + x^2};$$

check_deviations_for_spline(spline3_sub, runge);

plot([runge(x), spline3(runge)(x), MapleSpline3(x)], x = 0 ..1, color = [red, blue, green]);

$$runge := x \mapsto \frac{1}{1 + x^2}$$

$$deviation = 0.000995697876280954$$



>

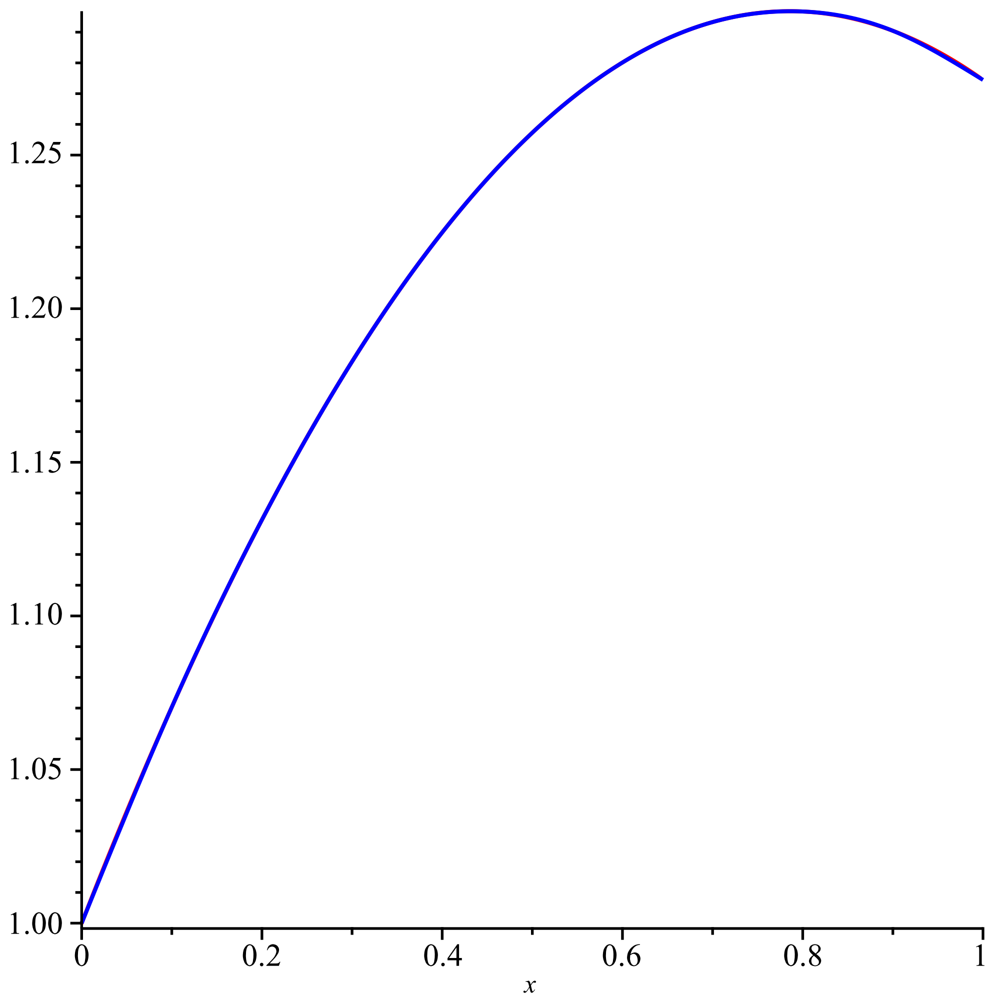
 # В статье "Cubic Spline Interpolation, Sky McKinley and Megan Levine" пишут: "While the fit is not perfect, it does closely approximate the function without a great degree of
 # divergence." Убедимся в этом на примере функции из статьи:

$f := x \rightarrow (\sin(x) + \cos(x))^{\frac{3}{4}};$

$check\_deviations\_for\_spline(spline3\_sub, f);$
$plot([f(x), spline3(f)(x)], x = 0..1, color = [red, blue]);$

$$f := x \mapsto (\sin(x) + \cos(x))^{3/4}$$

$$deviation = 0.000474009237593664$$



> # *Построим для этих же функций интерполяцию B-сплайнами:*

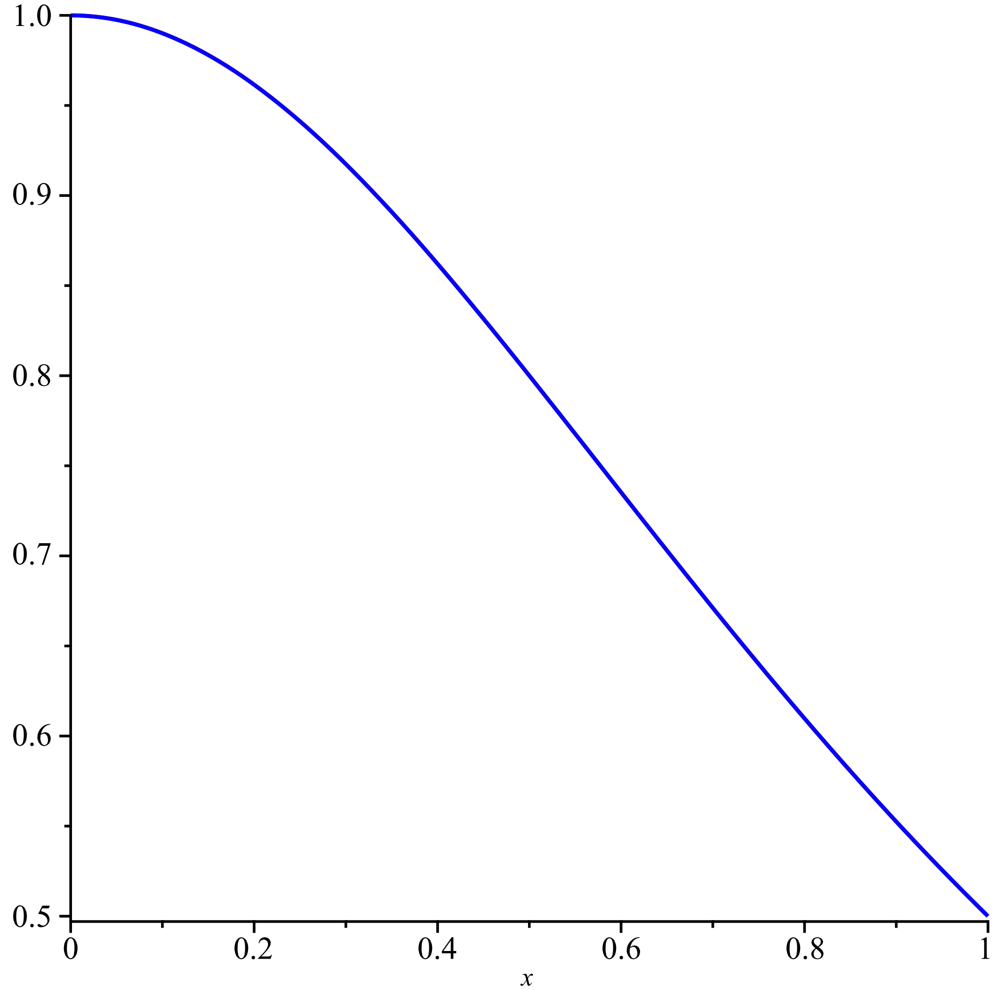
$runge := x \rightarrow \frac{1}{1 + x^2};$

$check\_deviations\_for\_spline(splineB\_sub, runge);$

$plot(\,[runge(x), splineB(runge)(x)\,], x = 0\,..1, color = [red, blue]);$

$$runge := x \mapsto \frac{1}{1 + x^2}$$

$$deviation = 0.0000468422$$

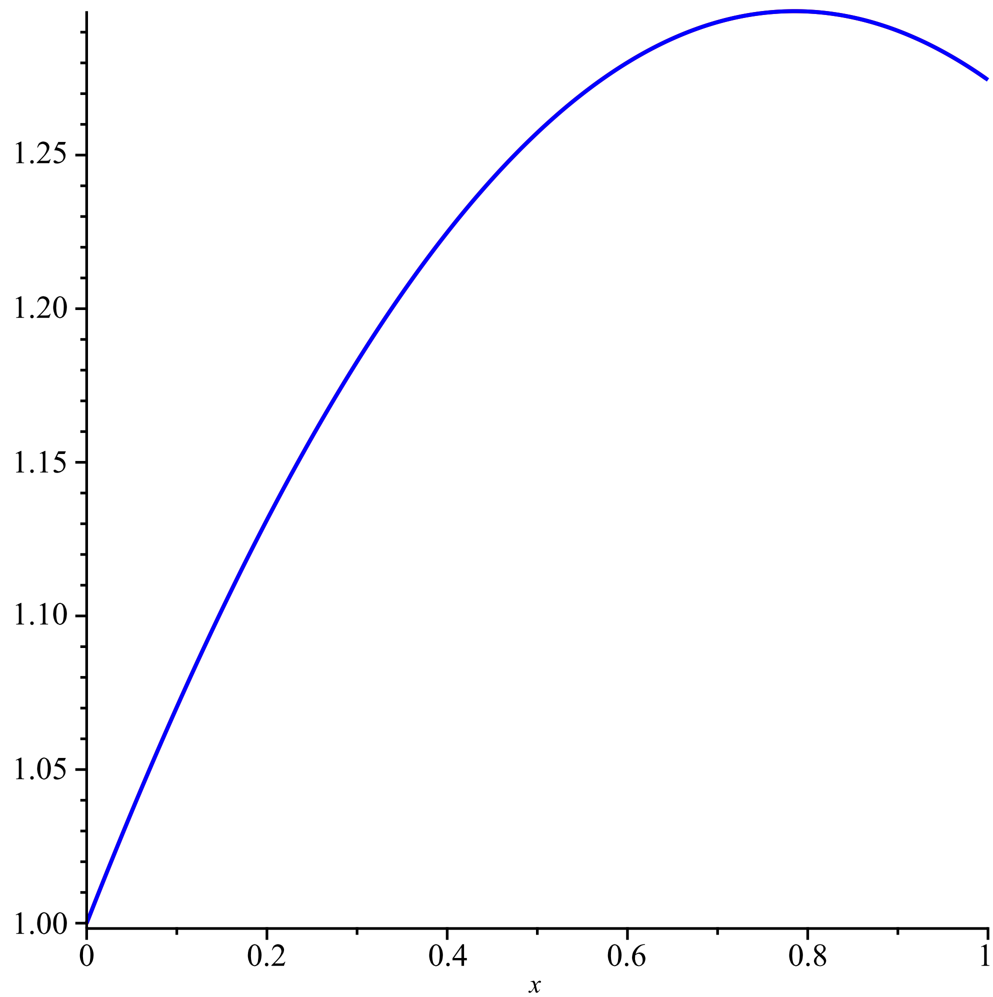


> $f := x \rightarrow (\sin(x) + \cos(x))^{\frac{3}{4}};$

$check\_deviations\_for\_spline(splineB\_sub, f);$
$plot([f(x), splineB(f)(x)], x = 0..1, color = [red, blue]);$

$$f := x \mapsto (\sin(x) + \cos(x))^{3/4}$$

$$deviation = 9.41 \times 10^{-7}$$



> # *Также рассмотрим пример функции из статьи "Quadratic B−Spline Curve Interpolation":*

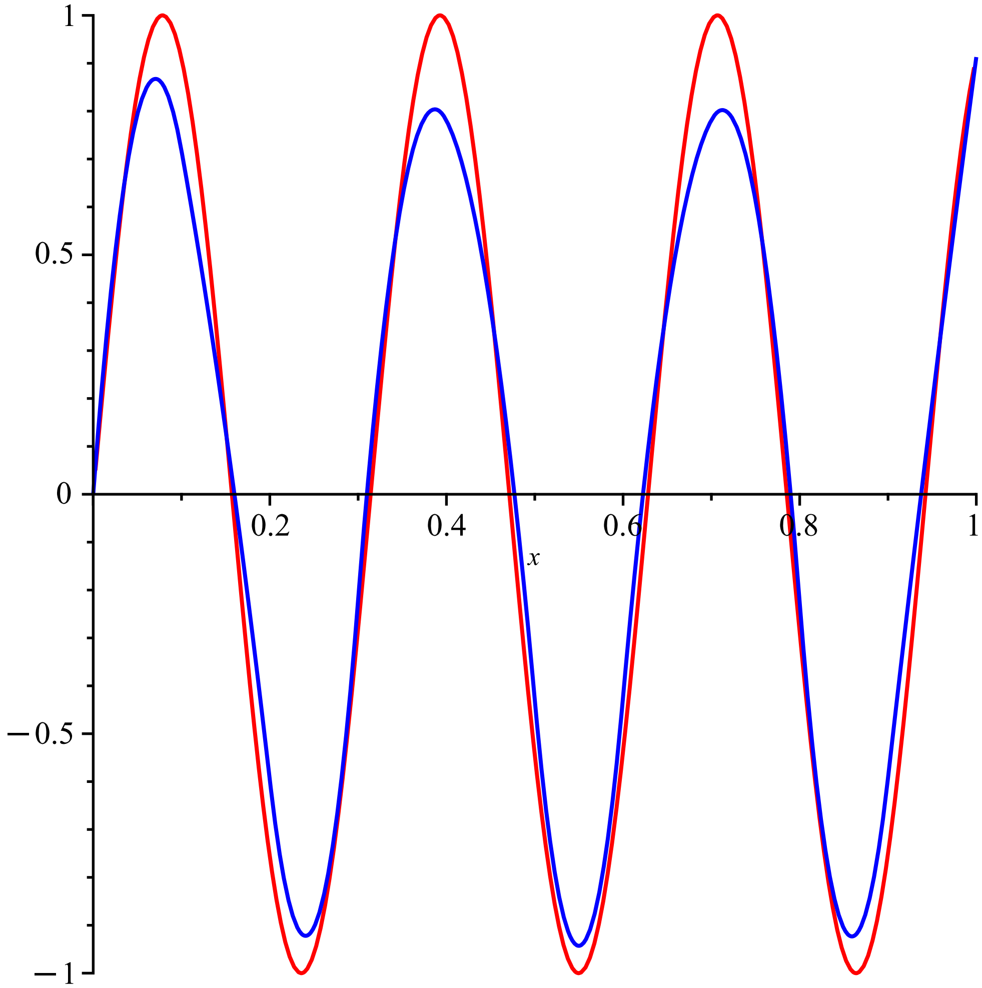
 # Как можно видеть, приближение с помощью квадратичных B-сплайнов оказалось хуже для данной функции. Авторы объясняют это следующим:

 # "The desired curve cannot be generated using an even−degree B−spline curve with joints at the interpolation points. This is because the segments of an even−degree B−spline curve are either concave−up or concave−down"

```
f := x → sin(20 x);
check_deviations_for_spline(splineB_sub, f);
plot([f(x), splineB(f)(x)], x = 0 ..1, color = [red, blue]);
```

$$f := x \mapsto \sin(20 \cdot x)$$

$$deviation = 0.2093370978$$



1
0.5
0
−0.5
−1
0.2
0.4
0.6
0.8
1
x