```
restart
interface(warnlevel=0);
with(LinearAlgebra) :
with(plots) :
with(ArrayTools) :


CubicSplineInterpolation_ := proc(points)
     #local n, h, vars_c, i,j, M, c, d, a_params, b_params, d_params,
    system_c, spline_3;

    n := numelems(points) − 1;  # Количество узлов интерполяции
     h := 0.1;  # Шаг между узлами
     # Вычисление коэффициентов векторов для кубических сплайнов
 vars_c := [seq(c[i], i=1 ..(n+1))];
 system_c := [c[1]=0];
 for i from 2 to n + 1 do
      if i = (n+1) then
            system_c := [seq(system_c[j],  j=1 ..(i−1)), c[i]= 0];

      else
            system_c := [seq(system_c[j],  j=1 ..(i−1)),  c[i−1]·h + 2
   ·2 h·c[i]  + c[i+1]·h = 6·((points[(i+1), 2] − points[i, 2])/h
   − (points[i, 2] −points[(i−1), 2])/h)];
      #system_c;
end if;
      end do;
      M := GenerateMatrix(system_c, vars_c, augmented = true);
     c := LinearSolve(M);
        a_params := seq(points[i, 2] , i=2 ..(n +1))  ;
        d_params := seq((c[i] −c[i−1])/h, i=2 ..(n +1)) ;
         b_params := seq((points[i, 2]−points[i−1, 2])/h + c[i]·h/3
   + c[i−1]·h/6, i=2 ..(n+1))  ;
 system_s := seq(a_params[i] + b_params[i]·(x−points[i + 1, 1])
```

```
+ c[i + 1]/2 · (x − points[i + 1, 1])^2 + d_params[i]/6 · (x − points[i
+ 1, 1])^3 , i = 1 ..n ) ;

spline_3 := piecewise(0 ≤ x < 0.1, system_s[1] , 0.1 ≤ x < 0.2,
  system_s[2] , 0.2 ≤ x < 0.3, system_s[3] , 0.3 ≤ x < 0.4,
  system_s[4] , 0.4 ≤ x < 0.5, system_s[5] , 0.5 ≤ x < 0.6,
  system_s[6] , 0.6 ≤ x < 0.7, system_s[7] , 0.7 ≤ x < 0.8,
  system_s[8] , 0.8 ≤ x < 0.9, system_s[9] , 0.9 ≤ x ≤ 1.0 ,
  system_s[10] , 0) ;

return spline_3;
end proc;
#step :=0.1;
#f:= x→ 1/(1+ 25 x^2);

#f := x → x^2;

#f:=x→(sin(x) + cos(x))^(3/4);

#x_ :=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];


x_ := [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0];
f_t := x→x^2;
xs := [seq([x_[i], f_t(x_[i] )], i = 1 ..11) ] ;
spline3 := CubicSplineInterpolation_(xs) ;
plot([f_t(x), CubicSplineInterpolation_(xs)(x)] , x=0 ..1, color
  =[red, blue] );
ReferenceCubicSpline := CubicSpline(xs, independentvar = z);
Draw(ReferenceCubicSpline);

  #построенный график кубического сплайна совпадает со сплайном из
  стандартной библиотеки maple.

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

```
checkCubicSpline := proc(func)
 local zero_one_grid, j, deviations, sp, i, smaller_xs_grid, k, diff;
    smaller_xs_grid, smaller_ys_grid, positive_difference;
zero_one_grid := seq(j, j = 0 .. 1, 0.1);
deviations := Array([ ]);
sp := x → spline(x);
 for i from 2 to 11 do
smaller_xs_grid := [seq(k, k = zero_one_grid[i − 1]
   .. zero_one_grid[i], 0.01)];
diff := x → abs(func(x) − spline3_sub(func, x));
deviations := Append(deviations, max(map(diff, smaller_xs_grid)));
 end do;
 return deviations;
 end proc;
```

$f2 := x \to \dfrac{1}{1+25x^2};$

```
xs := [seq([x_[i], f2(x_[i] )], i = 1 ..11) ] ;
 spline3 := CubicSplineInterpolation_(xs) ;
  #testVal:=subs(x=0.75, spline3(x)) : simplify(testVal);
plot([f2(x), CubicSplineInterpolation_(xs)(x)] , x=0 ..1,  color=[red,
    blue] );
```

#В статье "The Runge phenomenon and spatially variable shape parameters in RBF interpolation" **by** *Bengt Fornberg*
#and Julia Zuev" говорится

#про функцию Рунге, и её интерполяцию. Феномен Рунге заключается в том, что функция $f(x) = \dfrac{1}{25x^2+1}$ осциллирует с большой частотой на краях
# отрезка. Но как мы видим на графике, на краях отрезка [0, 1] эта функция не осциллирует,
и её получилось достаточно точно приблизить с
#помощью кубического сплайна.

```
  spline3 := CubicSplineInterpolation_(xs) ;
deviation := max(checkCubicSpline(f));
```

#судя по значению отклонения, функцию Рунге приближает весьма неплохо

```
f3 := x → (sin(x) + cos(x))^(3/4);
  xs := [seq([x_[i], f3(x_[i])], i = 1 ..11)];
 spline3 := CubicSplineInterpolation_(xs) ;
plot([f3(x), CubicSplineInterpolation_(xs)(x)] , x=0 ..1,  color=[red,
    blue] );
```

#В статьеCubic Spline Interpolation" by Sky McKinley and Megan Levine привододится пример приближения тригонометрической

#функции $(\sin(x)+\cos(x))^{\frac{3}{4}}$. Как видно из графика, её действительно можно хорошо приблизить с помощью кубического сплайна

```
BSplineInterpolation := proc(f)
  segment := 0 ..1;
  h := 0.1;
  n :=12;
  eps := 10^(-9);

  xs := [-2·eps, -eps,  seq(i,  i=segment, h), 1 + eps, 1 + 2·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);
 P := x→sum(lam(i)·B2(i, x), i=1 ..n);
 return P;
 end proc;

  f4 := x → sin(52 x);
 splineB := BSplineInterpolation(f4) ;
```

```
plot([f4(x), BSplineInterpolation(f4)(x) ], x =  0 ..1, color=[red,
   blue]);

   #B-сплайны плохо подходят для приблилижения тригонометрических
   фукций с большими коэффициентами перед x. Об этом говорится в
   статье #"Quadratic B-Spline Curve Interpolation " by Fuhua Cheng,
   Xuefu Wang and B. A Barsky", и действительно, это утверждение
   подтверждается
   #графиком построенного приближения.

 splineB := BSplineInterpolation(f2) ;
plot([f2(x), BSplineInterpolation(f2)(x) ], x =  0 ..1, color=[red,
   blue]);

   #Попробую приблизить функцию из первого примера для кубических
   сплайнов B-сплайном. Как видно из графика, B-сплайн даже лучше смог
   #приблизить функцию Рунге, подозреваю, что это

   #произошло по причине того, что набор базисных функций для
   квадратичного b-сплайна представляет из себя набор парабол,и они
   лучше подходят #для приближения функций подобного рода.
```

0

```
CubicSplineInterpolation_ := proc(points)
    local n, h, vars_c, i, system_c, j, M, c, a_params, d_params, b_params, system_s, spline_3;
    n := numelems(points) - 1;
    h := 0.1;
    vars_c := [seq(c[i], i=1 ..n + 1)];
    system_c := [c[1]=0];
    for i from 2 to n + 1 do
        if i=n + 1 then
            system_c := [seq(system_c[j], j=1 ..i - 1), c[i]=0]
        else
            system_c := [seq(system_c[j], j=1 ..i - 1), c[i - 1]*h + 4*h*c[i] + c[i + 1]*h=6
             * (points[i + 1, 2] - points[i, 2]) / h - 6* (points[i, 2] - points[i - 1, 2]) / h]
        end if
    end do;
    M := LinearAlgebra:-GenerateMatrix(system_c, vars_c, augmented=true);
    c := LinearAlgebra:-LinearSolve(M);
```

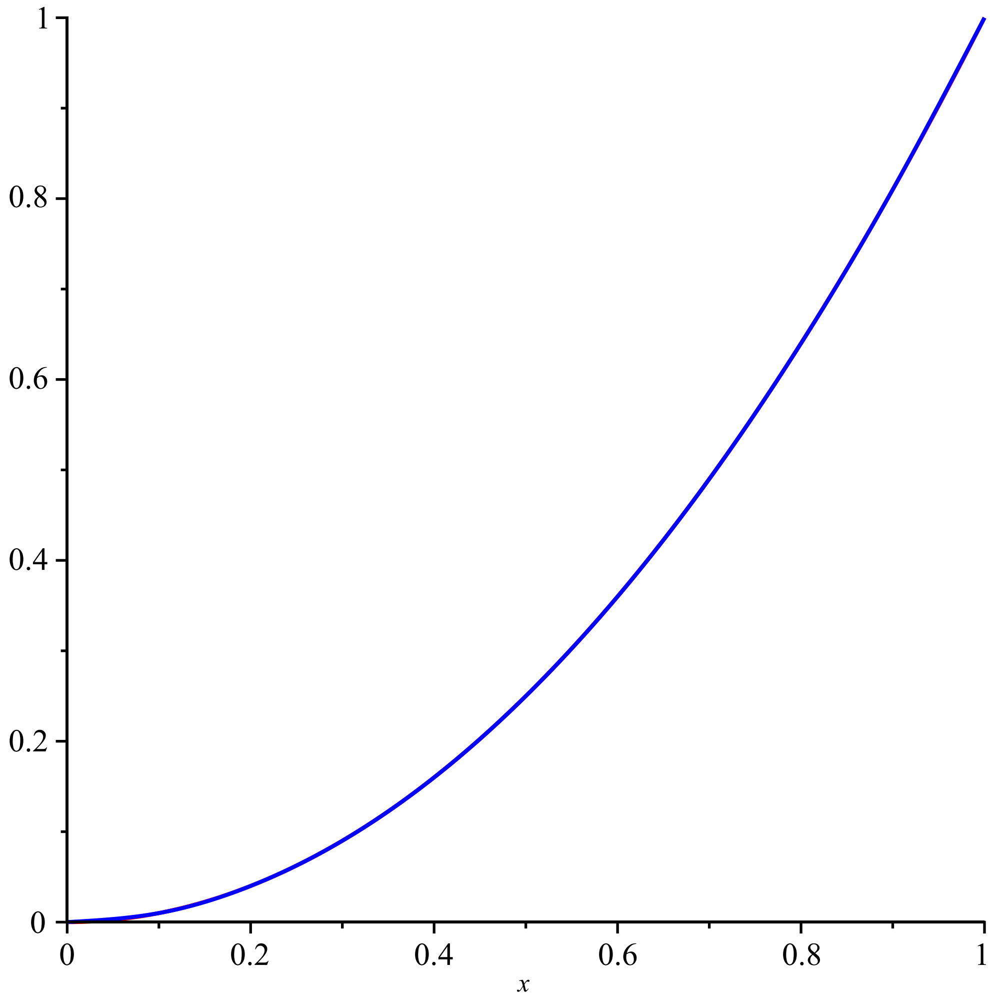
```
    a_params := seq(points[i, 2], i=2 ..n + 1);
    d_params := seq((c[i] - c[i - 1]) / h, i=2 ..n + 1);
    b_params := seq((points[i, 2] - points[i - 1, 2]) / h + 1 / 3 * h * c[i] + 1 / 6 * c[i - 1] * h, i=2
    ..n + 1);
    system_s := seq(a_params[i] + b_params[i] * (x - points[i + 1, 1]) + 1 / 2 * c[i + 1]
    * (x - points[i + 1, 1])^2 + 1 / 6 * d_params[i] * (x - points[i + 1, 1])^3, i=1 ..n);
    spline_3 := piecewise(0 <=x and x < 0.1, system_s[1], 0.1 <=x and x < 0.2, system_s[2], 0.2
    <=x and x < 0.3, system_s[3], 0.3 <=x and x < 0.4, system_s[4], 0.4 <=x and x < 0.5,
    system_s[5], 0.5 <=x and x < 0.6, system_s[6], 0.6 <=x and x < 0.7, system_s[7], 0.7 <=x
    and x < 0.8, system_s[8], 0.8 <=x and x < 0.9, system_s[9], 0.9 <=x and x <= 1.0, system_s
    [10], 0);
    return spline_3
end proc
```

$$x\_ := [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$$

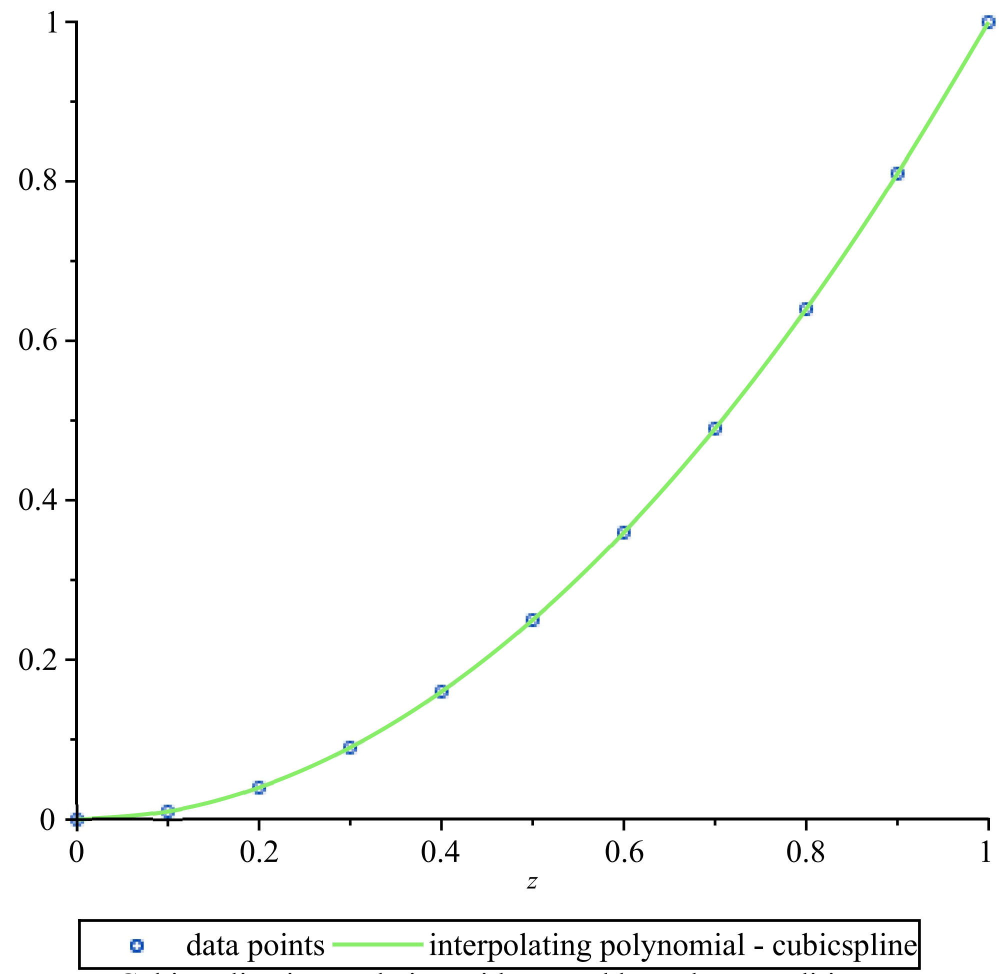
$$f\_t := x \mapsto x^2$$

$$xs := [[0, 0], [0.1, 0.01], [0.2, 0.04], [0.3, 0.09], [0.4, 0.16], [0.5, 0.25], [0.6, 0.36], [0.7, 0.49], [0.8, 0.64], [0.9, 0.81], [1.0, 1.00]]$$

$$spline3 := \begin{cases} -0.00845303867403315 + 0.184530386740332\, x + 1.26795580110497\, (x - 0.1)^2 + 4.22651933 \\ -0.0408287292817680 + 0.404143646408840\, x + 0.928176795580110\, (x - 0.2)^2 - 1.13259668 \\ -0.0896685082872928 + 0.598895027624309\, x + 1.01933701657459\, (x - 0.3)^2 + 0.30386740 \\ -0.160110497237569 + 0.800276243093923\, x + 0.994475138121547\, (x - 0.4)^2 - 0.08287292 \\ -0.250000000000000 + 1.\, x + 1.00276243093923\, (x - 0.5)^2 + 0.0276243093922649\, ( \\ -0.359834254143646 + 1.19972375690608\, x + 0.994475138121547\, (x - 0.6)^2 - 0.02762430 \\ -0.490773480662983 + 1.40110497237569\, x + 1.01933701657459\, (x - 0.7)^2 + 0.08287292 \\ -0.636685082872928 + 1.59585635359116\, x + 0.928176795580110\, (x - 0.8)^2 - 0.30386740 \\ -0.823922651933702 + 1.81546961325967\, x + 1.26795580110497\, (x - 0.9)^2 + 1.13259668 \\ -0.942265193370166 + 1.94226519337017\, x - 4.22651933701657\, (x - 1.0)^3 \\ 0 \end{cases}$$




$ReferenceCubicSpline \coloneqq POLYINTERP([[0, 0], [0.1, 0.01], [0.2, 0.04], [0.3, 0.09], [0.4, 0.16], [0.5, 0.25], [0.6, 0.36], [0.7, 0.49], [0.8, 0.64], [0.9, 0.81], [1.0, 1.00]], independentvar = z, INFO)$




Cubic spline interpolation with natural boundary conditions.

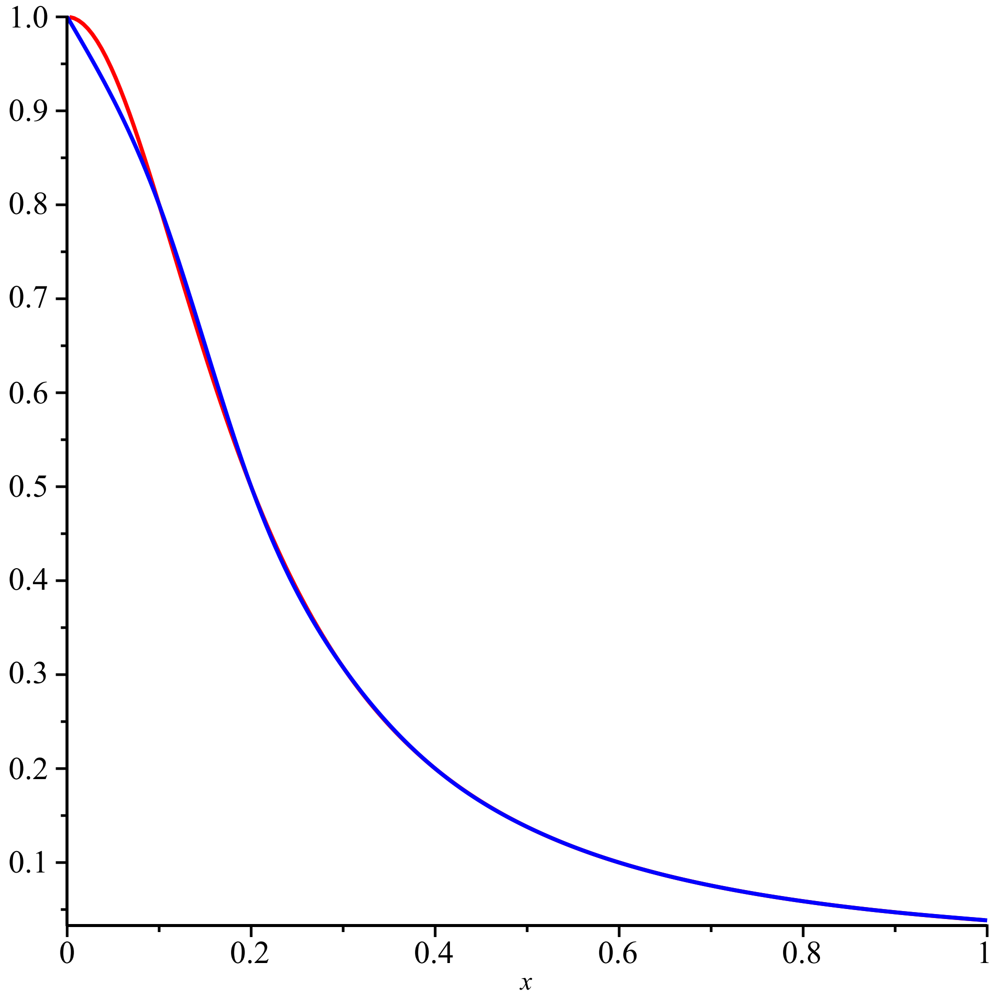
```
spline3_sub := proc(f, b) local q1; q1 := subs(x = b, spline3(f) (x)); return eval(q1) end proc

checkCubicSpline := proc(func)
    local zero_one_grid, j, deviations, sp, i, smaller_xs_grid, k, diff;
    smaller_xs_grid, smaller_ys_grid, positive_difference;
    zero_one_grid := seq(j, j = 0 ..1, 0.1);
    deviations := Array([ ]);
    sp := x→spline(x);
    for i from 2 to 11 do
        smaller_xs_grid := [seq(k, k = zero_one_grid[i − 1]..zero_one_grid[i], 0.01)];
        diff := x→abs(func(x) − spline3_sub(func, x));
        deviations := ArrayTools:-Append(deviations, max(map(diff, smaller_xs_grid)))
    end do;
    return deviations
end proc
```

$$f2 := x \mapsto \frac{1}{1 + 25 \cdot x^2}$$

$xs := [[0, 1], [0.1, 0.8000000000], [0.2, 0.5000000000], [0.3, 0.3076923077], [0.4, 0.2000000000],$
$[0.5, 0.1379310345], [0.6, 0.1000000000], [0.7, 0.07547169811], [0.8, 0.05882352941], [0.9,$
$0.04705882353], [1.0, 0.03846153846]]$

$$spline3 := \begin{cases} 1.06621036694889 - 2.66210366948892\,x - 9.93155504233383\,(x-0.1)^2 - 33.10518347444 \\ 1.03652743135775 - 2.68263715678877\,x + 9.72622016933533\,(x-0.2)^2 + 65.52591737223 \\ 0.720665849606796 - 1.37657847302265\,x + 3.33436666499253\,(x-0.3)^2 - 21.30617834780 \\ 0.524419580314912 - 0.811048950787279\,x + 2.32092856069455\,(x-0.4)^2 - 3.37812701432 \\ 0.373962994247448 - 0.472063919494896\,x + 1.06892175222928\,(x-0.5)^2 - 4.17335602821 \\ 0.280417222739881 - 0.300695371233136\,x + 0.644763730388326\,(x-0.6)^2 - 1.41386007280 \\ 0.214725979107460 - 0.198934687139228\,x + 0.372843111217413\,(x-0.7)^2 - 0.90640206390 \\ 0.169911527791294 - 0.138859997976618\,x + 0.227903779742023\,(x-0.8)^2 - 0.48313110491 \\ 0.135269226108870 - 0.0980115584209663\,x + 0.180580615814494\,(x-0.9)^2 - 0.15774387975 \\ 0.118415035299517 - 0.0799534968395169\,x - 0.601935386048314\,(x-1.0)^3 \\ 0 \end{cases}$$



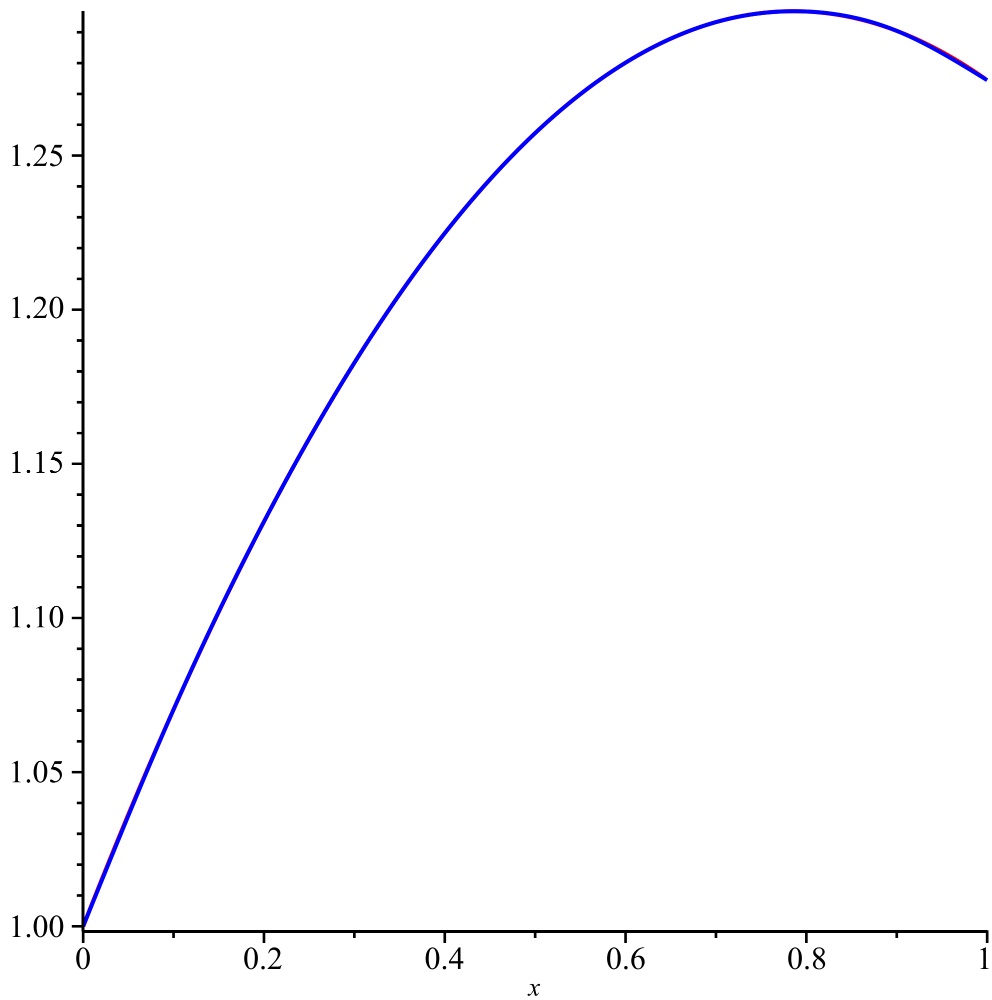

$$spline3 := \begin{cases} 1.06621036694889 - 2.66210366948892\,x - 9.93155504233383\,(x-0.1)^2 - 33.1051834744 \\ 1.03652743135775 - 2.68263715678877\,x + 9.72622016933533\,(x-0.2)^2 + 65.5259173722 \\ 0.720665849606796 - 1.37657847302265\,x + 3.33436666499253\,(x-0.3)^2 - 21.306178347 \\ 0.524419580314912 - 0.811048950787279\,x + 2.32092856069455\,(x-0.4)^2 - 3.37812701432 \\ 0.373962994247448 - 0.472063919494896\,x + 1.06892175222928\,(x-0.5)^2 - 4.1733560282 \\ 0.280417222739881 - 0.300695371233136\,x + 0.644763730388326\,(x-0.6)^2 - 1.413860072 \\ 0.214725979107460 - 0.198934687139228\,x + 0.372843111217413\,(x-0.7)^2 - 0.906402063 \\ 0.169911527791294 - 0.138859997976618\,x + 0.227903779742023\,(x-0.8)^2 - 0.483131104 \\ 0.135269226108870 - 0.0980115584209663\,x + 0.180580615814494\,(x-0.9)^2 - 0.157743879 \\ 0.118415035299517 - 0.0799534968395169\,x - 0.601935386048314\,(x-1.0)^3 \\ 0 \end{cases}$$

$$deviation := 1.$$

$$f3 := x \mapsto (\sin(x) + \cos(x))^{3/4}$$

$$xs := [[0, 1], [0.1, 1.070316628], [0.2, 1.131259881], [0.3, 1.182784619], [0.4, 1.224821026], [0.5, 1.257293962], [0.6, 1.280135571], [0.7, 1.293293473], [0.8, 1.296735866], [0.9, 1.290454241], [1.0, 1.274464067]]$$

$$spline3 := \begin{cases} 1.00395819307197 + 0.663584349280314\,x - 0.593728960795289\,(x-0.1)^2 - 1.97909653598 \\ 1.01915952349622 + 0.560501787518901\,x - 0.437096656818843\,(x-0.2)^2 + 0.52210767992 \\ 1.04225014980678 + 0.468448230644083\,x - 0.483438911929340\,(x-0.3)^2 - 0.15447418370 \\ 1.07580517003809 + 0.372539639904769\,x - 0.475646995463798\,(x-0.4)^2 + 0.02597305488 \\ 1.11895721213158 + 0.276673499736842\,x - 0.483014406215470\,(x-0.5)^2 - 0.02455803583 \\ 1.17201394431128 + 0.180202711147863\,x - 0.481693479674324\,(x-0.6)^2 + 0.00440308847 \\ 1.23554278302981 + 0.0825009856717069\,x - 0.495323775087235\,(x-0.7)^2 - 0.04543431804 \\ 1.30649410906775 - 0.0121978038346903\,x - 0.451664119976737\,(x-0.8)^2 + 0.14553218370 \\ 1.39745229829965 - 0.118886730332946\,x - 0.615225145005816\,(x-0.9)^2 - 0.54520341676 \\ 1.45487331183353 - 0.180409244833527\,x + 2.05075048335272\,(x-1.0)^3 \\ 0 \end{cases}$$



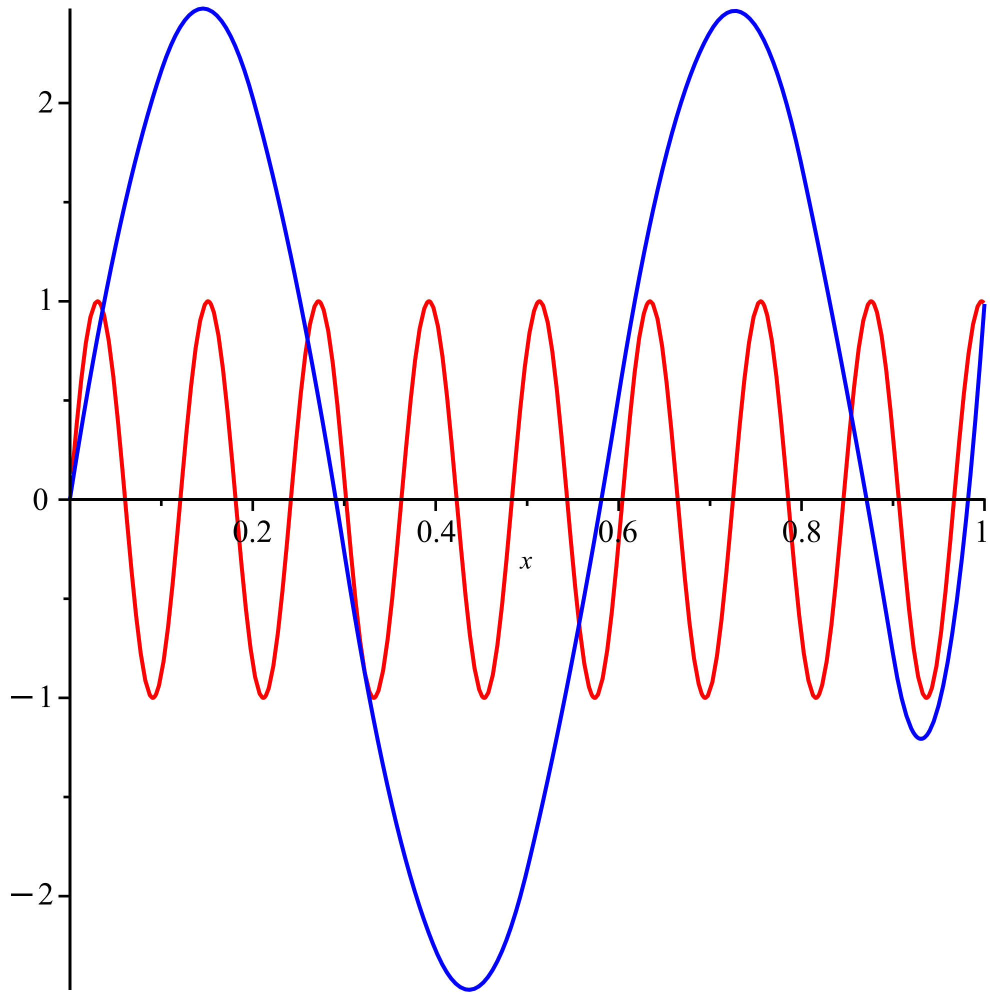

```
BSplineInterpolation := proc(f)
    local segment, h, n, eps, xs, i, ys, lam, B0, B1, B2, P;
    segment := 0 ..1;
    h := 0.1;
    n := 12;
    eps := 1 / 1000000000;
    xs := [ - 2 * eps, - eps, seq(i, i = segment, h), eps + 1, 1 + 2 * eps];
    ys := [ f(0), f(0), seq(f(i), i = segment, h), f(1), f(1) ];
    lam := j→piecewise(j = 1, f(xs[1]), 1 < j and j < n, - 1 / 2 * f(xs[j + 1]) + 2 * f(1 / 2 * xs[j
     + 1] + 1 / 2 * xs[j + 2]) - 1 / 2 * f(xs[j + 2]), j = n, f(xs[n + 1]));
    B0 := (i, x)→piecewise(xs[i] <= x and x < xs[i + 1], 1, 0);
    B1 := (i, x)→(x - xs[i]) * B0(i, x) / (xs[i + 1] - xs[i]) + (xs[i + 2] - x) * B0(i + 1, x) / (xs[i
     + 2] - xs[i + 1]);
    B2 := (i, x)→(x - xs[i]) * B1(i, x) / (xs[i + 2] - xs[i]) + (xs[i + 3] - x) * B1(i + 1, x) / (xs[i
     + 3] - xs[i + 1]);
    P := x→sum(lam(i) * B2(i, x), i = 1 ..n);
```

**return** $P$

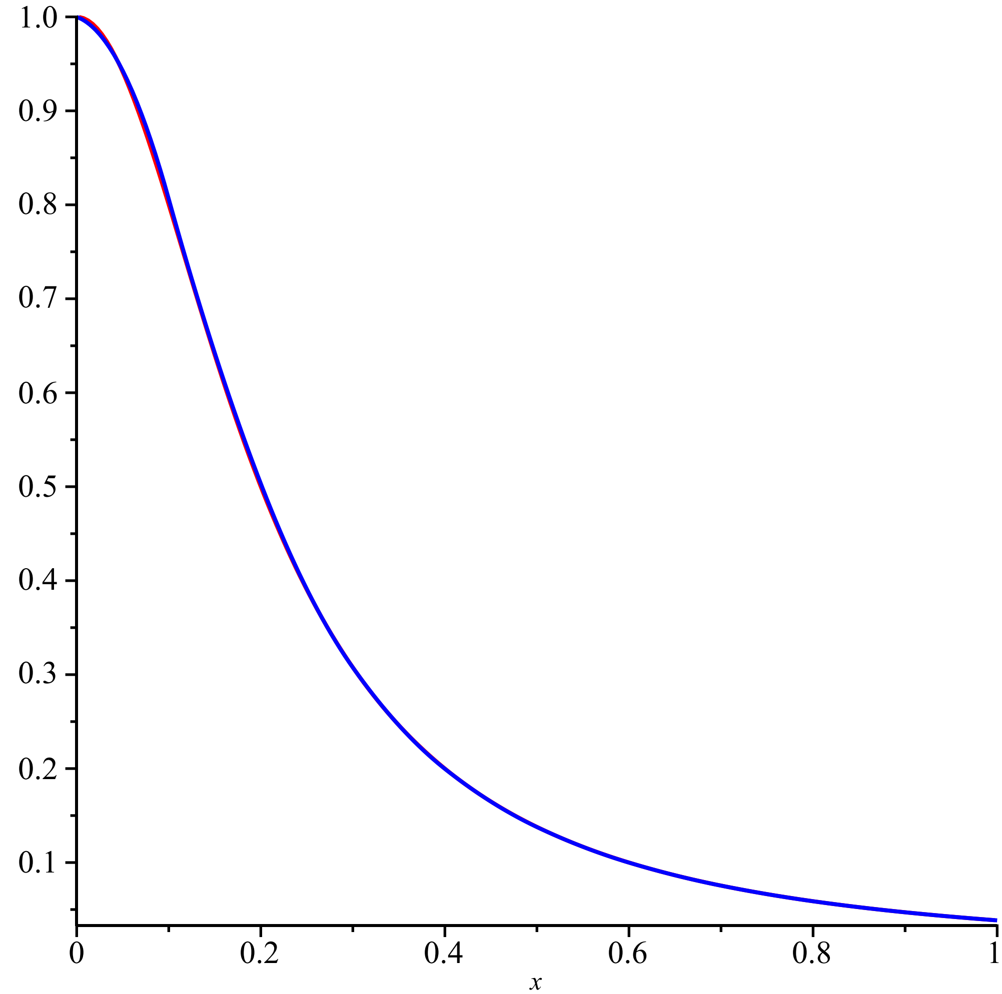
**end proc**

$$f4 := x \mapsto \sin(52 \cdot x)$$

$$splineB := P$$




$$splineB := P$$


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.4
0.6
0.8
1
x