```
> restart;
> h := 1/10 :
> n := 10 :
```

Построим B-сплайн

```
> BSPL := proc(t)
  local i;
  local B;
  local eps := 10^(-8);
  local m := n + 2
  local x := [-2·eps, -eps, seq(i·h, i=0..n), 1 + eps, 1 + 2·eps];
  local y := [f(0), f(0), seq(f(i·h), i=0..n), f(1), f(1)];
  local c := i → piecewise(i=1, y_1, 1 < i < m, 1/2 (-y_(i+1) + 4 f((x_(i+1) + x_(i+2))/2) - y_(i+2)), i=m,
      y_(m+1));
  B_0 := (i, t)→ { 1   x_i ≤ t < x_(i+1) ; 0   otherwise } ;
  B_1 := (i, t)→ (t - x_i)/(x_(i+1) - x_i) ·B_0(i, t) + (x_(i+2) - t)/(x_(i+2) - x_(i+1)) ·B_0(i + 1, t) ;
  B_2 := (i, t)→ (t - x_i)/(x_(i+2) - x_i) ·B_1(i, t) + (x_(i+3) - t)/(x_(i+3) - x_(i+1)) ·B_1(i + 1, t) ;
  return sum(c(i)·B_2(i, t), i=1..m) ;
  end proc:
```

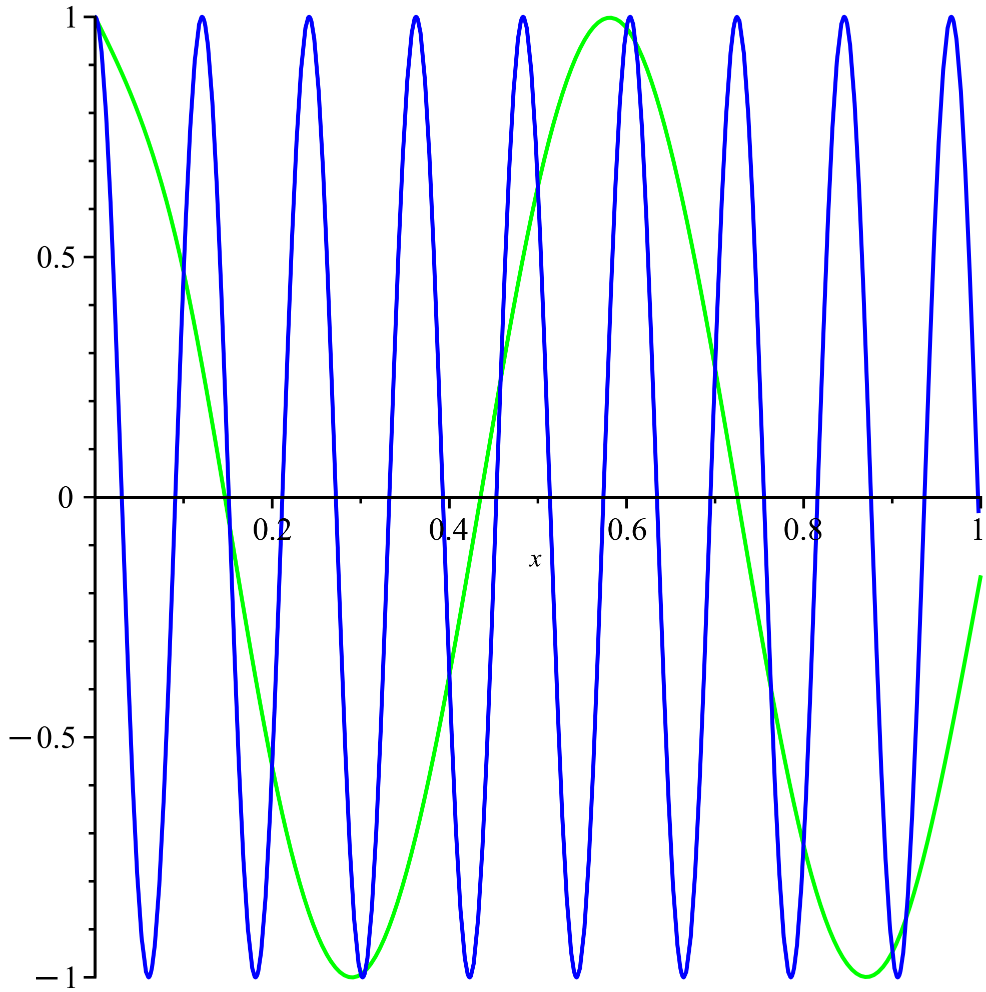
Построим Кубический сплайн

```
> CSPL := proc(t)
  local x := Array(0..n, i→ i·h);
  local eqs := [c_0 = 0, c_n = 0];
  local a, b, c, d, i;
  local parts;
  local spl;
  local s := (t, i) → a[i] + b[i]·(t - x[i]) + c[i]/2 (t - x[i])^2 + d[i]/6 (t - x[i])^3 ;

  for i from 1 to n - 1 do
    eqs := [op(eqs), c_(i-1)·h + 2(h + h)·c_i + c_(i+1)·h = 6·((f(x[i + 1]) - f(x[i]))/h
       - (f(x[i]) - f(x[i - 1]))/h)];
  end do;
  assign(fsolve(eqs));

  for i from 1 to n do
```

```
    a[i] := f(x[i]);
    d[i] := (c[i] - c[i-1]) / h;
    b[i] := (f(x[i]) - f(x[i-1])) / h + c[i]·h / 3 + c[i-1]·h / 6;
  end do;

  parts := [ ];

  for i from 1 to n do
      parts := [op(parts), x[i - 1] ≤ t ≤ x[i], s(t, i)];
  end do;
  spl := piecewise(op(parts));
  return spl(t);

  end proc:
> computeError := proc(f, interpolator)
  local segment := 0 ..1;
  local h := 0.01;
  local i;
  local xs := [seq(i, i = segment, h)];
  local diff := x → abs(interpolator(x) − f(x));
  local errors := map(diff, xs);
  return evalf(max(errors));
  end proc:
>
```

Проверим, что полученные сплайны индентичны стандартным

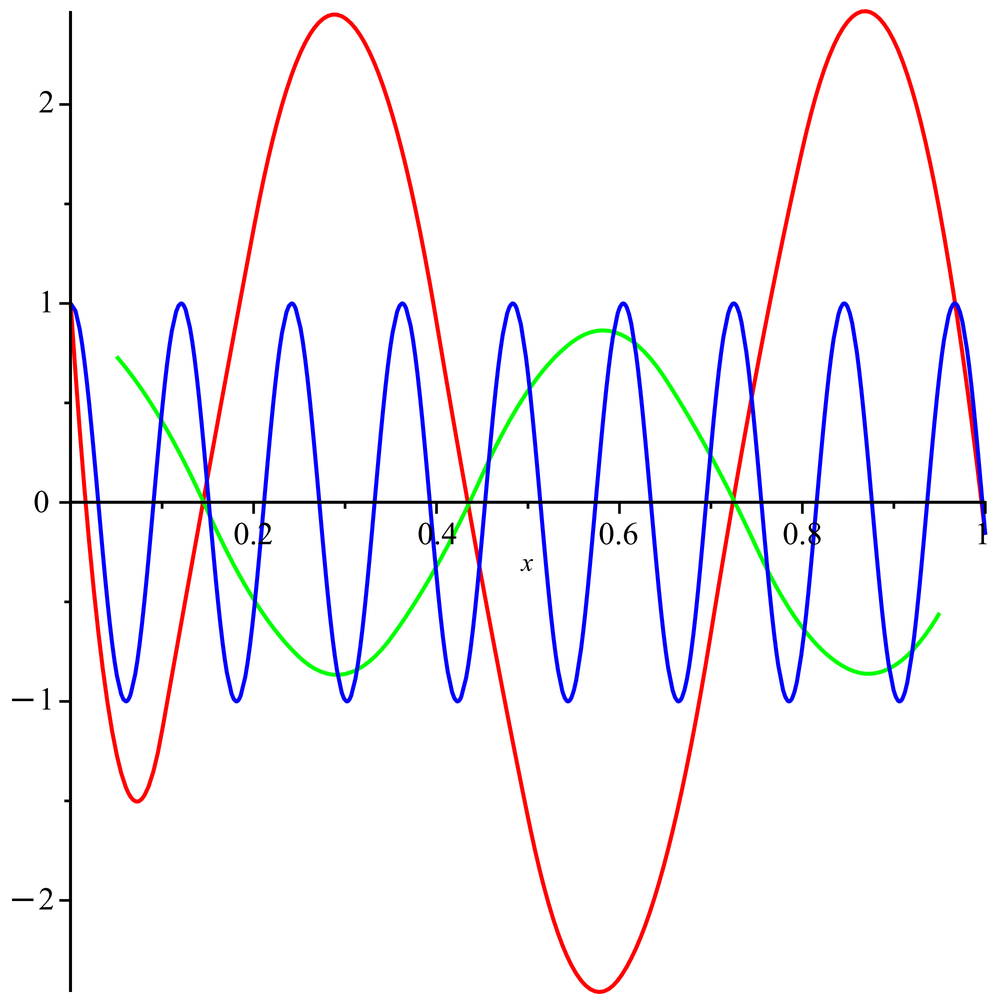
```
> with(CurveFitting) :
> MBSPL(x) := BSplineCurve(
   [−2·eps, −eps, seq(i, i = 0 ..1, 0.1), 1 + eps, 1 + 2·eps],
   [f(0), f(0), seq(f(i), i = 0 ..1, 0.1), f(1), f(1)],
   x, order = 3 ) :

> MCSPL(x) := Spline([seq(i, i = 0 ..1, 0.1)], [seq(f(i), i = 0 ..1, 0.1)], x, degree = 3) :

> f(t) := cos(52· t) :
> plot([CSPL(x), MCSPL(x), f(x)], x = 0 ..1, color = [red, green, blue])
```



> computeError(f, *CSPL*) − computeError(f, MCSPL)

$$7.82676590205256 \times 10^{-10} \quad (1)$$

> plot([$BSPL(x)$, $MBSPL(x)$, $f(x)$], $x = 0..1$, $color = [red, green, blue]$)



Разница обусловлена тем, что c[i] в стандартной библиотеке отличаются от книжных. Оба варианта являются B-сплайнами по построению.

 # Рубрика эксперименты

 # В https://drlvk.github.io/nm/section-drawbacks-spline-interpolation.html утверждается, что сплайны:

 # 1) На монотонных данных могут давать немонотонные результаты

 # 2) На всюду положительных данных могут выдавать отрицательные

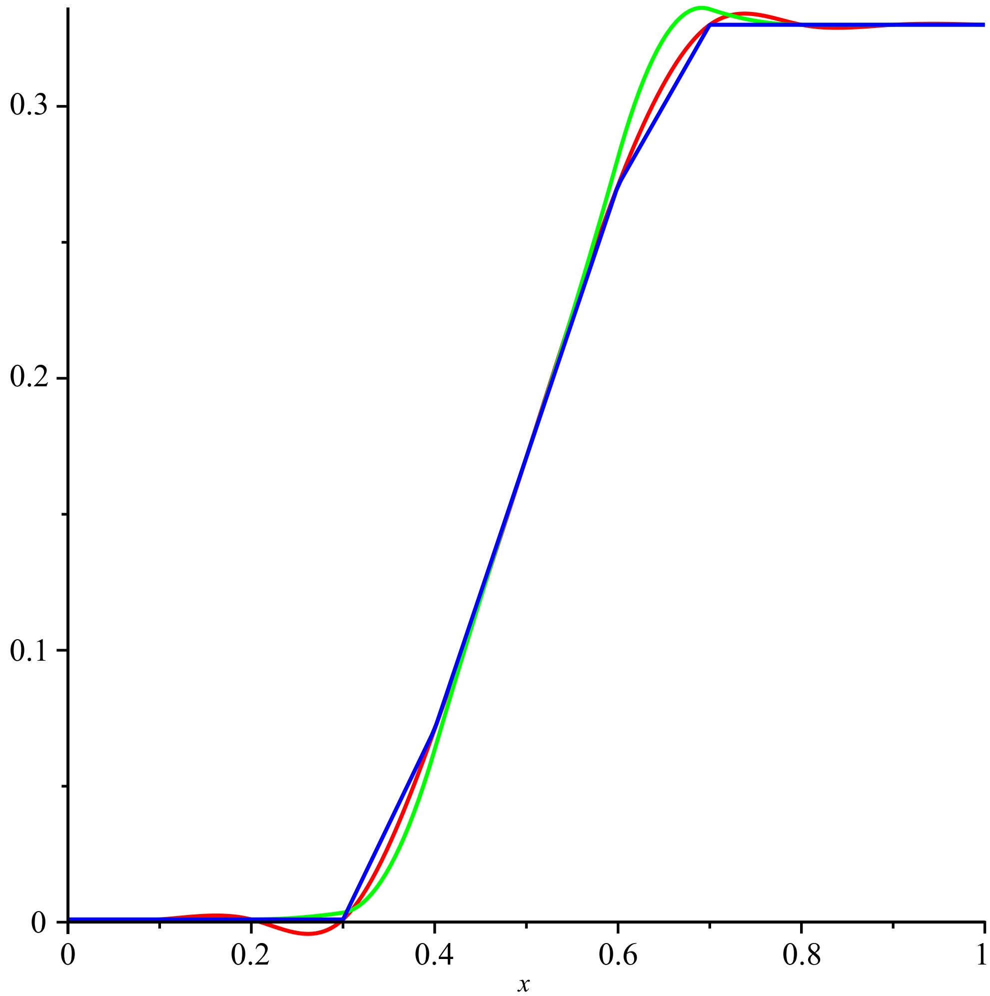
 # Поэтому их приминение ограничено для областей, где это критично. В таких случаях вместо сплайнов используют линейную интерполяцию.

1) Рассмотрим монотонную положительную функцию (заметим, что точки сетки можно гладко соединить):

> $f(t) := piecewise(0 \le t \le 0.33, 0.001, 0.33 < t < 0.66, 0.001 + (t - 0.33), 0.66 \le t \le 1, 0.33)$

$$f := t \mapsto \begin{cases} 0.001 & 0 \le t \le 0.33 \\ -0.329 + t & 0.33 < t < 0.66 \\ 0.33 & 0.66 \le t \le 1 \end{cases} \quad \textbf{(2)}$$

```
> with(Interpolation):
> LIN(t) := Interpolate([seq(i, i=0 .. 1, 0.1)], [seq(f(i), i=0 .. 1, 0.1)], method=linear)(t):
> plot([CSPL(x), BSPL(x),   LIN(x) ], x=0 ..1, color=[red, green, blue] );
```



Как видим нарушается не только монотонность, но и положительность у сплайнов. Линейная интерполяция сохраняет положительность и монотонность

```
> [computeError(f, CSPL), computeError(f, BSPL), computeError(f, LIN)]
```

[0.0158900484, 0.0117100000, 0.0236000000000000] **(3)**

Но ценой большей ошибки.

```
>
```