```
> restart;
n := 10 :
step := 1/n :
grid := [seq((i - 1)*1/n, i = 1 ..n + 1)];
with(LinearAlgebra) :
```

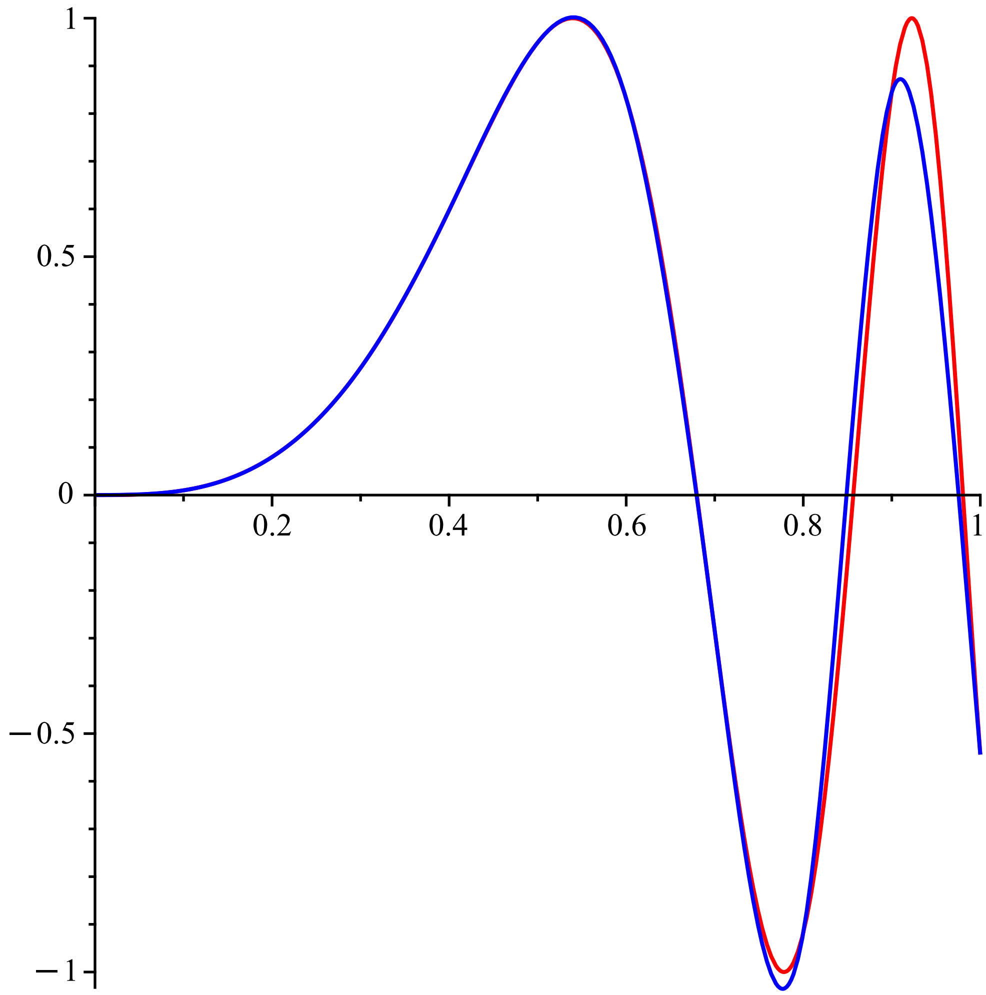
$$grid := \left[0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1\right] \quad \textbf{(1)}$$

```
> # Cubic splines
cubSpline := proc(f)
    local yx, matr_init, matr, vect, sol, a, b, c, blyadina, cub_proc;
    yx := [seq(f(grid[i]), i = 1 ..n + 1)];
    matr_init := (i, j)→if i = j and 1 < i and i < n + 1 then
        4·step
    elif abs(i - j) = 1 then
        step
    elif i = j then
        1
    else
        0
    end if:;
    matr := Matrix(n + 1, n + 1, matr_init);
    vect := Vector(n + 1, i→if i = 1 or i = n + 1 then

        0

    else

        6·(yx[i + 1] - yx[i])/step - 6·(yx[i] - yx[i - 1])/step

    end if);
    sol := LinearSolve(matr, vect);
    a := Array(1 ..n, i→f(grid[i + 1]));
    b := Array(1 ..n, i→(yx[i + 1] - yx[i])/step + 1/3·step·sol[i + 1] + 1/6·step·sol[i]);
    c := Array(1 ..n, i → (sol[i + 1] - sol[i])/step);
    blyadina := (i, x) → a[i] + b[i]·(x - grid[i + 1]) + 1/2·sol[i + 1]·(x - grid[i
    + 1])^2 + 1/6·c[i]·(x - grid[i + 1])^3;
    cub_proc := proc(x)
        for i to n do
```

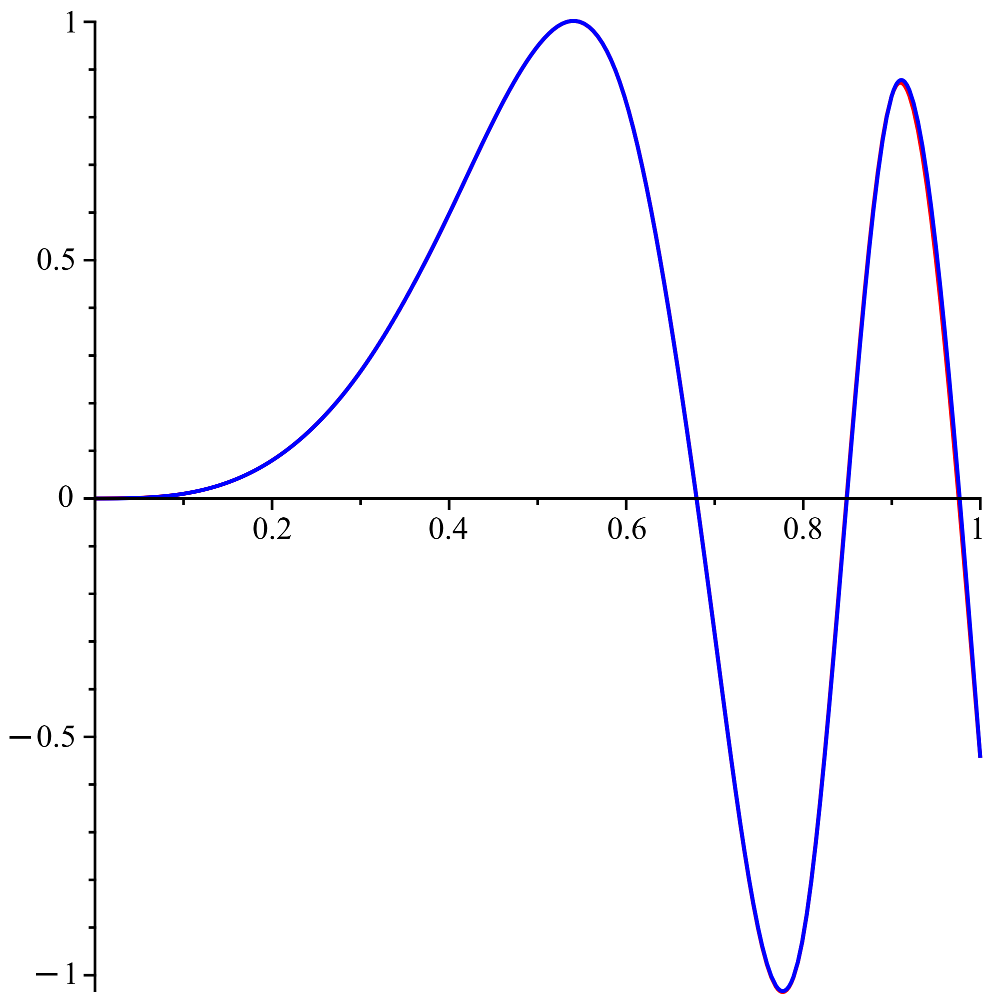
```
            if grid[i] ≤ x and x ≤ grid[i + 1] then
                return blyadina(i, x)
            end if
        end do
    end proc:;
    return f → cub_proc(f)
end proc:;
```

```
> f := x→sin(10 x^3);
  plot([f, cubSpline(f) ], 0 ..1, color = [red, blue]);
```

$$f := x \mapsto \sin(10 \cdot x^3)$$



```
> with (CurveFitting) :;
  CubicSplineMaple := x→Spline([seq(i, i = 0 ..1, 0.1)], [seq(f(i), i = 0 ..1, 0.1)], x, degree
      = 3) :;
  plot([cubSpline(f), CubicSplineMaple], 0 ..1, color = [red, blue]);
```



> # B splines

$eps := 10^{-9}$:;

$grid := [-2 \cdot eps, -eps, seq(i, i = 0 ..1, step), 1 + eps, 1 + 2 \cdot eps]$:;

$k := n + 2$:;

$yx := [f(0), f(0), seq(f(i), i = 0 ..1, step), f(1), f(1)]$:;

$$c := i \rightarrow piecewise\left(i = 1, yx[1], 1 < i < k, -\frac{yx[i+1]}{2} + 2 \cdot f\left(\frac{grid[i+1]}{2} + \frac{grid[i+2]}{2}\right) - \frac{yx[i+2]}{2}, i = k, \; yx[k+1]\right) :;$$

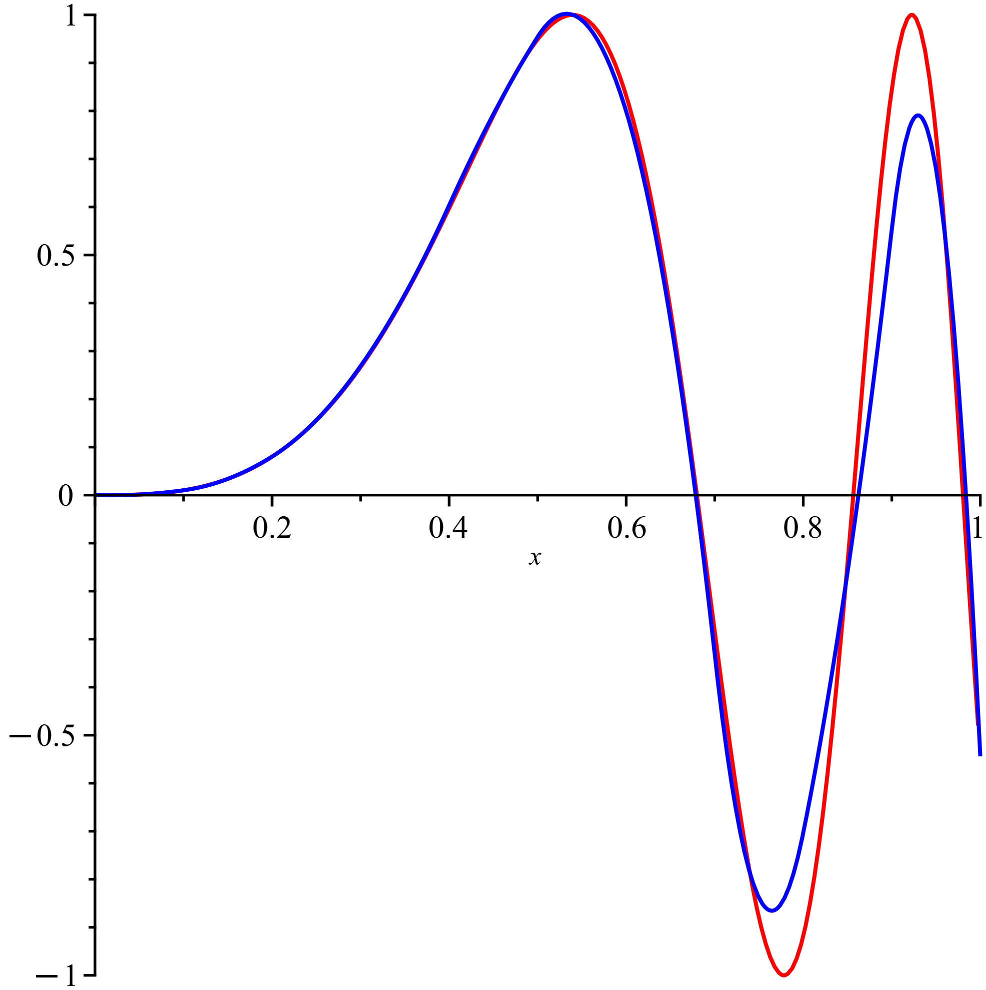
$B[0] := (i, t) \rightarrow piecewise(grid[i] \leq t < grid[i+1], 1, 0)$:;

$$B[1] := (i, t) \rightarrow \frac{(t - grid[i]) \cdot B[0](i, t)}{(grid[i+1] - grid[i])} + \frac{(grid[i+2] - t) \cdot B[0](i+1, \; t)}{(grid[i+2] - grid[i+1])} :;$$

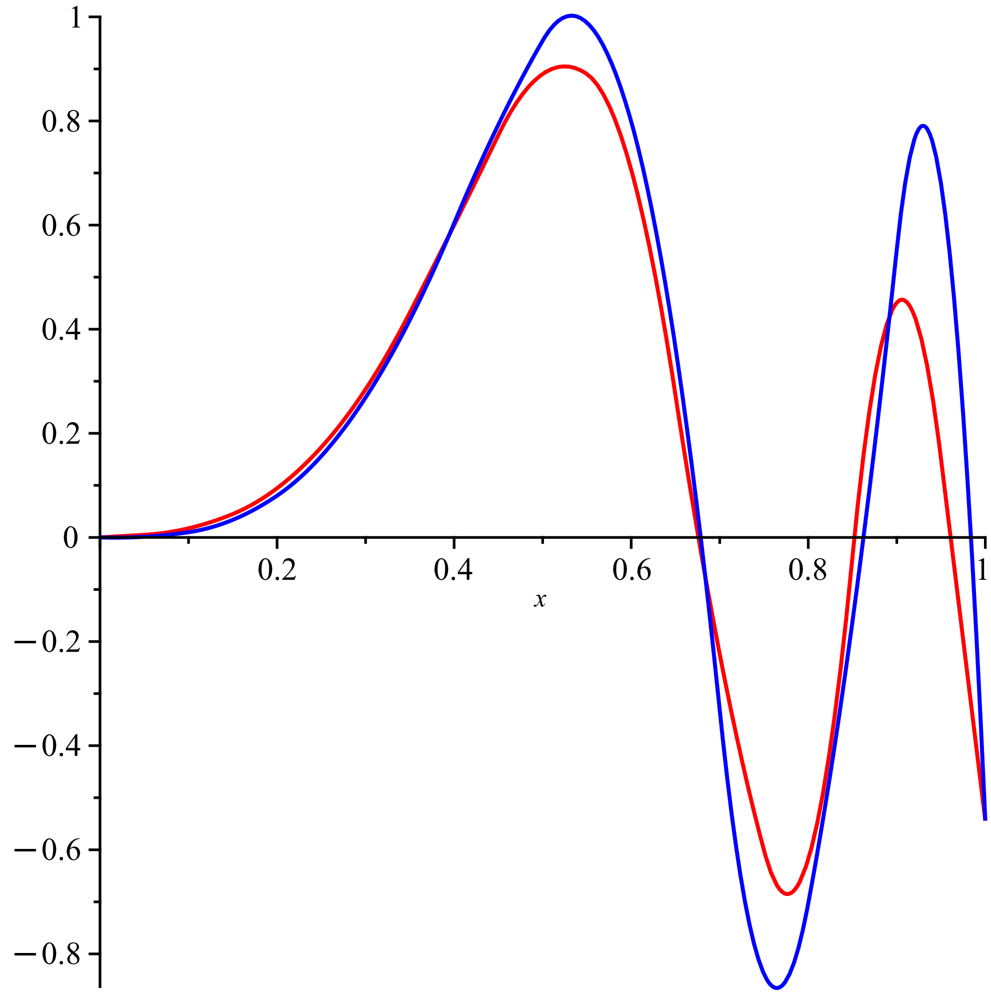
$$B[2] := (i, t) \rightarrow \frac{(t - grid[i]) \cdot B[1](i, t)}{(grid[i+2] - grid[i])} + \frac{(grid[i+3] - t) \cdot B[1](i+1, \; t)}{(grid[i+3] - grid[i+1])} :;$$

$Bspline := x \rightarrow sum(c(i) \cdot B[2](i, x), i = 1 ..k)$:;

> $plot([f(x), \; Bspline(x)], x = 0 ..1, color = [red, blue])$;



> $with(CurveFitting)$:;
$BSplineMaple := x \rightarrow BSplineCurve([-2 \cdot eps, -eps, seq(i, i = 0 ..1, step), 1 + eps, 1 + 2 \cdot eps],$
$[f(0), f(0), seq(f(i), i = 0 ..1, step), f(1), f(1)], x, order = 3)$:;
$plot([BSplineMaple(x), Bspline(x)], x = 0 ..1, color = [red, blue]);$



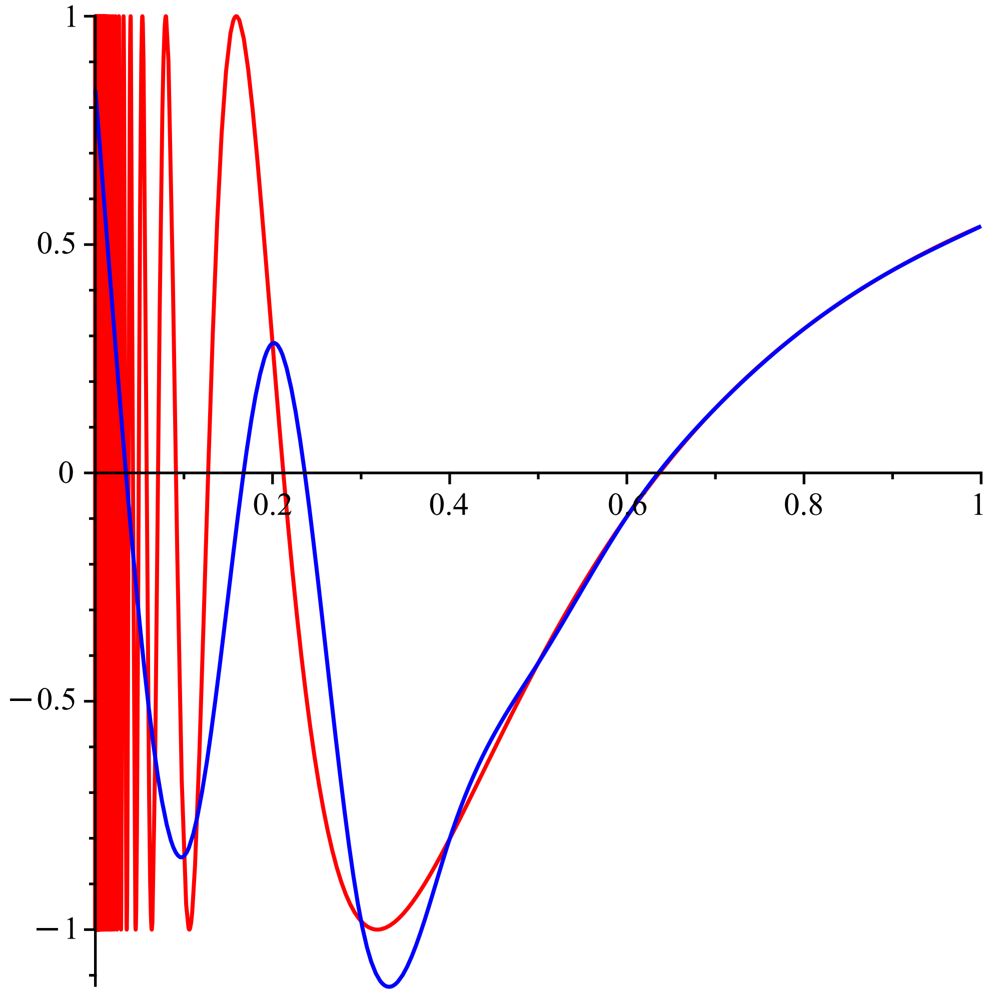

```
> # Calculate error
  calc_err := proc(f, g)
  local err, err_arr, step, points;
  step := 1/100 :;
  points := [seq(i, i = 0 ..1, step)] :;
  err := x→abs(f(x) − g(x));
  err_arr := [seq(err(point), point in points)];
  return max(err_arr);
  end proc:
```

> # *Эксперимент*

Рассмотрим отрывок из книги "Numerical Analysis" *Richard L. Burden, J. Douglas Faires,* в *главе про сплайны описан достаточно интересный эффект, связанный* с *тем, что кубические спайны* "не очень хорошо" (*далее будет показано, что значит эта фраза*) *аппроксимируют осциллирующие функции . Проверим этот факт.*

```
> f := x→cos(1/(x + 10^(−9)));
  plot([f, cubSpline(f)], 0 ..1, color = [red, blue]);
  evalf[10](calc_err(f, cubSpline(f)));
```

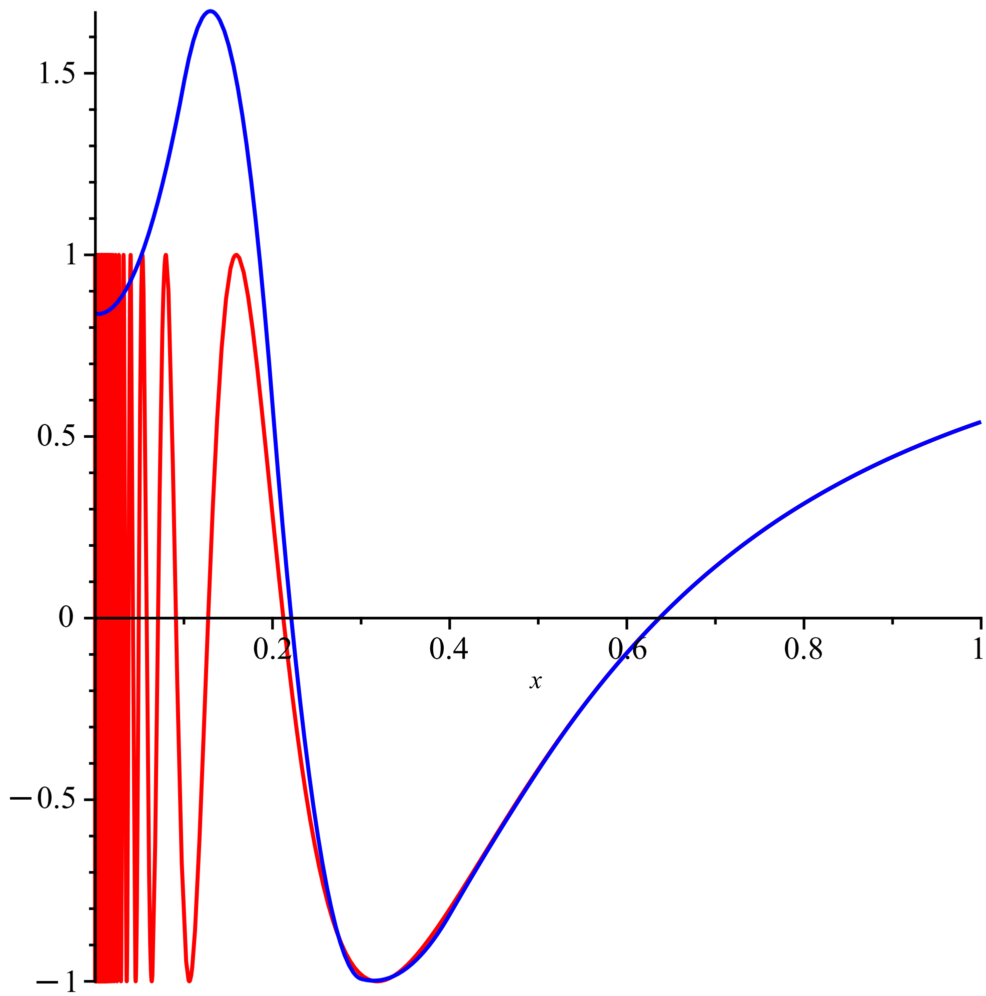
$$f := x \mapsto \cos\left(\frac{1}{x + \frac{1}{1000000000}}\right)$$



$$1.763407209 \quad \textbf{(2)}$$

> "Не очень хорошо" − *это ошибка больше* 1, *что* и *иллюстрирует данный пример, но при это на неосцеллирующем участке ошибка достаточна мала* (*меньше* 0.001), *что подтверждает факт из книги. Проверим то же самое для* б *сплайнов.*

> *plot*([*f* (*x*), *Bspline*(*x*)], *x* = 0 ..1, *color* = [*red*, *blue*]);
> *evalf* [10](*calc_err*(*f*, *Bspline*));



2.528916449 **(3)**

> *Аппроксимация* Б *сплайнами работает еще хуже, но подтверждает факт, представленный* в *книге.*