```
with(LinearAlgebra) : with(ArrayTools) :
n := 10
```

$$n := 10 \tag{1}$$

```
f := x→sin(x);
xs := evalf( Array( 1..( n + 1 ), i→ (i − 1)/n ) );
h := Array( 1..n, i→(xs[i + 1] − xs[i]) );
ys := evalf( Array( 1..(n + 1) , i→f(xs[i]) ) );
```

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

$$xs := \left[\begin{array}{cccccc} 0. & 0.1000000000 & 0.2000000000 & 0.3000000000 & 0.4000000000 & 0.500000000\cdots \end{array}\right]$$

$$h := \left[\begin{array}{cccccc} 0.1000000000 & 0.1000000000 & 0.1000000000 & 0.1000000000 & 0.1000000000 & (\cdots \end{array}\right]$$

$$ys := \left[\begin{array}{cccccc} 0. & 0.09983341665 & 0.1986693308 & 0.2955202067 & 0.3894183423 & 0.47942553\cdots \end{array}\right] \tag{2}$$

```
my_spline := proc(f)
 fill_matrix := proc(i, j)
   if (i = j ) then return 2·(h[i ] + h[i + 1])
   elif (abs(i−j) = 1) then return h[i + 1]
  else return 0
   end if
end proc;
 A := Matrix(n − 1, n − 1, fill_matrix );
 v := Vector( n − 1, i→ 3( (ys[i + 2] − ys[i + 1])/h[i + 1] − (ys[i + 1] − ys[i])/h[i] ) );
 c := evalf(LinearSolve(A, v));
 c := Concatenate(1, 0, c, 0);

 a := evalf(seq(ys[i], i = 1..n + 1));
 d := evalf( seq( (c[i] − c[i − 1])/(3· h[i − 1]), i = 2..n + 1 ) );

 b := evalf( seq( (ys[i] − ys[i − 1])/h[i − 1] − (c[i]·h[i − 1])/3 − (2· c[i − 1]·h[i − 1])/3, i = 2 .. n
     + 1 ) ) ;

 S := seq( a[i] + b[i]·(x − xs[i]) + c[i]/1 ·(x − xs[i])^2 + d[i]/1 ·(x − xs[i])^3, i = 1
```

$$..n\Big);$$

$$xs\_test := evalf\left(Array\left(1..(100 + 1),\ i \to \frac{i-1}{100} \right) \right);$$

S :

return *piecewise*(
$0 \le x < 0.1, S[1],$
$0.1 \le x < 0.2, S[2],$
$0.2 \le x < 0.3, S[3],$
$0.3 \le x < 0.4, S[4],$
$0.4 \le x < 0.5, S[5],$
$0.5 \le x < 0.6, S[6],$
$0.6 \le x < 0.7, S[7],$
$0.7 \le x < 0.8, S[8],$
$0.8 \le x < 0.9, S[9],$
$0.9 \le x < 1, S[10]$
);
end proc:

Warning, (in my_spline) `fill_matrix` is implicitly declared local
Warning, (in my_spline) `A` is implicitly declared local
Warning, (in my_spline) `v` is implicitly declared local
Warning, (in my_spline) `c` is implicitly declared local
Warning, (in my_spline) `a` is implicitly declared local
Warning, (in my_spline) `i` is implicitly declared local
Warning, (in my_spline) `d` is implicitly declared local
Warning, (in my_spline) `b` is implicitly declared local
Warning, (in my_spline) `S` is implicitly declared local
Warning, (in my_spline) `xs_test` is implicitly declared local

$f\_splined := my\_spline(f)$

$$f\_splined := \begin{cases} 0.999999536721184\,x - 0.166537022118392\,x^3 \\ 0.000333074010903447 + 0.995003426390966\,x - 0.0499611066355177\,(x - 0.1000000000)^2 - \\ 0.00265599435700914 + 0.980066682214954\,x - 0.0994063334579291\,(x - 0.2000000000)^2 - \\ 0.00892014292523463 + 0.955333545915885\,x - 0.147925029532766\,(x - 0.3000000000)^2 - \\ 0.0209905506513970 + 0.921069479121507\,x - 0.194715638411007\,(x - 0.4000000000)^2 - \\ 0.0406512913009570 + 0.877548494598086\,x - 0.240494206823204\,(x - 0.5000000000)^2 - \\ 0.0693661881083111 + 0.825460475486148\,x - 0.280385984296175\,(x - 0.6000000000)^2 - \\ 0.109155844179875 + 0.764374061457321\,x - 0.330478155992096\,(x - 0.7000000000)^2 + 0. \\ 0.158594647952346 + 0.698451803684567\,x - 0.328744421735441\,(x - 0.8000000000)^2 - 0 \\ 0.229741053376032 + 0.615095395804409\,x - 0.504819657066140\,(x - 0.9000000000)^2 + \end{cases}$$

$spline(xs, ys, x);$

$$\begin{cases} 0.999999536619901\,x - 0.166537011990144\,x^3 \\ 0.000333074023980279 + 0.995003426260197\,x - 0.0499611035970433\,(x - 0.1000000000)^2 - 0.16481744 \\ 0.00265599433213806 + 0.980066682339310\,x - 0.0994063356118309\,(x - 0.2000000000)^2 - 0.16172897 \\ 0.00892014293523091 + 0.955333545882564\,x - 0.147925028955630\,(x - 0.3000000000)^2 - 0.15596869 \\ 0.0209905506478258 + 0.921069479130435\,x - 0.194715638565650\,(x - 0.4000000000)^2 - 0.15259522 \\ 0.0406512913021531 + 0.877548494595694\,x - 0.240494206781767\,(x - 0.5000000000)^2 - 0.13297259 \\ 0.0693661881079264 + 0.825460475486789\,x - 0.280385984307278\,(x - 0.6000000000)^2 - 0.16697390 \\ 0.109155844179996 + 0.764374061457149\,x - 0.330478155989122\,(x - 0.7000000000)^2 + 0.00577911417 \\ 0.158594647952309 + 0.698451803684614\,x - 0.328744421736234\,(x - 0.8000000000)^2 - 0.58691745 \\ 0.229741053376043 + 0.615095395804396\,x - 0.504819657065941\,(x - 0.9000000000)^2 + 1.68273219 \end{cases}$$

$$errors\_arr\_spline := \mathrm{Array}\left(0..100,\ \mathrm{i} \to \mathrm{abs}\left(\mathrm{eval}\left(f\_splined,\ \mathrm{x} = \frac{\mathrm{i}}{100}\right) - \mathrm{evalf}\left(\mathrm{f}\left(\frac{\mathrm{i}}{100}\right)\right)\right)\right):;$$

```
abs_error_spline := evalf(max(errors_arr_spline));
print(abs_error_spline);
```

$$abs\_error\_spline := 0.8414709848$$

$$0.8414709848 \qquad \textbf{(5)}$$

```
my_b_spline := proc(f)
 local n := 12, h := 0.1;
  xs := [seq(i, i = -0.2 ..1.2, h) ];
 c := (i) -> piecewise(i = 1, f(xs[1]), 1 < i and i < n, -1/2 ·f(xs[i + 1]) + 2
    ·f(1/2 ·xs[i + 1] + 1/2 ·xs[i + 2]) - 1/2 ·f(xs[i + 2]), i = n, f(xs[n + 1]));
 B0 := (i, t) -> piecewise(t < xs[i] , 0,  xs[i] ≤ t and t < xs[i + 1], 1, xs[i + 1]
    ≤ t, 0);

 B1 := (i, t) -> (t - xs[i])·B0(i, t)/(xs[i + 1] - xs[i]) + (xs[i + 2] - t)·B0(i + 1, t)/(xs[i + 2] - xs[i + 1]);
 B2 := (i, t) -> (t - xs[i])·B1(i, t)/(xs[i + 2] - xs[i]) + (xs[i + 3] - t)·B1(i + 1, t)/(xs[i + 3] - xs[i + 1]);
 P := t -> sum(c(i)·B2(i, t), i = 1 .. n);
 return P;
 end proc:
Warning, (in my_b_spline) `xs` is implicitly declared local
Warning, (in my_b_spline) `i` is implicitly declared local
Warning, (in my_b_spline) `c` is implicitly declared local
Warning, (in my_b_spline) `B0` is implicitly declared local
Warning, (in my_b_spline) `B1` is implicitly declared local
Warning, (in my_b_spline) `B2` is implicitly declared local
Warning, (in my_b_spline) `P` is implicitly declared local
 f_aprx := my_b_spline(f) :;
errors_arr := Array(0 ..100, i-> abs(evalf(f_aprx(i/100)) - evalf(f(i/100)))) :;
abs_error := evalf(max(errors_arr) ) :;
print(abs_error);
```

0.07431385029 **(6)**