1. **Interpolation.** Approximate functions $f_1(t)$ and $f_2(t)$, given in selected points in Table 1, using polynomial, rational function and cubic spline interpolation.

t	$f_1(t)$	$f_2(t)$
0	2.00	0.00
1	1.21	-0.04
2	-0.23	-0.35
3	-0.79	-1.14
4	-0.16	-1.37
5	0.54	0.19
6	0.37	1.46
7	-0.29	0.96
8	-0.46	0.25
9	0.04	0.03
10	0.43	0.00

Table 1: Functions $f_1(t)$ and $f_2(t)$.

Compute the values of the interpolating functions at $n = 200$ evenly spaced points, including the start and end points. For your report, plot the interpolating polynomial and rational functions together with the data points in one figure for both f_1 and f_2 separately. Also, plot the splines in single figure as a parametric plot, *i.e* as x–coordinates of the curve use the spline for function $f_1(t)$ and the one for $f_2(t)$ as the y–coordinate. That's three figures all together. Discuss the quality of the interpolations and possible short-comings of the used methods in your report.

2. **Derivatives.** Write a subroutine that calculates the first derivatives of a function using the three point formula. The values of the function have been tabulated at evenly spaced abscissae and the derivatives are to be stored in another table at the same points. The syntax should be

```
deriv3(h, y, dy, n)
```

`h` is the tabulation step length, that is, the distance between adjacent points.

`y` is an array of length `n`, storing the values of the function.

`dy`, array of length `n`, the calculated derivatives.

`n`, number of tabulation points.

Test your routine by tabulating and calculating the derivatives of the function

$$f(x) = \frac{\sin(x^2)}{x}, \tag{1}$$

in the range $-4 < x < 4$. Study the effect of increasing the number of tabulation points by comparing your results to the known derivative.

$$f'(x) = 2\cos(x^2) - \frac{\sin(x^2)}{x^2}. \tag{2}$$

void `polint(float xa[], float ya[], int n, float x, float *y, float *dy)` Evaluates the polynomial interpolating given points at a point

`float xa[]`, input Array of x–coordinates

`float ya[]`, input Array of y–coordinates

`int n`, input Number of points

`float x`, input Point where to evaluate the polynomial

`float *y`, output Pointer to the value of the interpolating function

`float *dy`, output Pointer to an error estimate

void `ratint(float xa[], float ya[], int n, float x, float *y, float *dy)` Evaluates the rational function interpolating given points at a point

`float xa[]`, input Array of x–coordinates

`float ya[]`, input Array of y–coordinates

`int n`, input Number of points

`float x`, input Point where to evaluate the rational function

`float *y`, output Pointer to the value of the interpolating function

`float *dy`, output Pinter to an error estimate

void `spline(float xi[], float yi[], int n, float y1p, float ynp, float y2[])` Constructs a table of second derivatives for use with cubic spline interpolation

`float xi[]`, input Array of x–coordinates

`float yi[]`, input Array of y–coordinates

`int n`, input Size of arrays `xi` and `yi`

`float y1p`, input Derivative of the function at start point. If $> 10^{30}$, assume natural spline (first derivative chosen so that second derivative is zero)

`float ynp`, input Derivative of the function at end point. If $> 10^{30}$, assume natural spline (first derivative chosen so that second derivative is zero)

`float y2[]`, output Array of the second derivatives

void `splint(float xi[], float yi[], float y2[], int n, float x, float *y)` Evaluates the spline

`float xi[]`, input Array of x–coordinates

`float yi[]`, input Array of y–coordinates

`float y2[]`, output Array of the second derivatives, as constucted by `spline`–routine

`int n`, input Size of arrays `xi`, `yi` and `y2`

`float x`, input Point where spline is evaluated

`float *y`, output Pointer to the value of the spline