Please write your family and given names and **underline** your family name on the front page of your paper.

1. It can be shown (you don't need to show it) that

$$B(x) \equiv \frac{1}{2}\begin{cases} x^2 & 0 \le x \le 1 \\ 1 + 2(x-1) - 2(x-1)^2 & 1 \le x \le 2 \\ (3-x)^2 & 2 \le x \le 3 \\ 0 & \text{elsewhere} \end{cases} \tag{1}$$

is a quadratic B-spline in $\mathbb{R}$, defined with respect to the knots 0, 1, 2, 3.

(a) [*3 points*] Consider the set of knots $\{x_i,\ i = 0, \cdots, n\}$, with $a \equiv x_0 < x_1 < \cdots < x_{n-1} < x_n \equiv b$. It is known that a quadratic spline in $[a, b]$ defined with respect to $n+1$ knots has $n+2$ degrees of freedom (free parameters). For simplicity, assume that the knots are equidistant, i.e. $x_i \equiv a + ih$, $i = 0, \cdots, n$, with $h \equiv (b-a)/n$.
Based on $B(x)$ of (1) and using appropriate mappings define a set of quadratic B-spline basis functions $\{B_i(x),\ i = 0, \cdots, n+1\}$ with respect to the knots $\{x_i,\ i = 0, \cdots, n\}$. That is, each $B_i(x)$ has the same shape with $B(x)$ after appropriate mapping of the intervals [0, 1], [1, 2] and [2, 3] to $[x_{i-2}, x_{i-1}]$, $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$, respectively.

(b) [*5 points*] Using MATLAB, in one graph, plot $B_i(x)$, $i = 0, \cdots, n+1$ in the interval [0, 1] (i.e. for $a = 0$ and $b = 1$), once for $n = 4$ and once more for $n = 8$ (i.e. once for $h = 0.25$ and once more for $h = 0.125$). Sample each $B_i(x)$ on 81 points in [0, 1] to draw the plot. Hand-in a hard-copy of the source code of your script and of your plot.

(c) [*2 points*] Which B_j's are non-zero at some knot x_i and what are their values; Do these values depend on n?

(d) [*2 points*] Let $t_i = (x_{i-1} + x_i)/2$, $i = 1, \cdots, n$. Which B_j's are non-zero at some midpoint t_i and what are their values; Do these values depend on n?

(e) [*3 points*] Which B'_j's are non-zero at some knot x_i and what are their values; Do these values depend on n? (Note: B'_j denotes the first derivative of B_j.)

2. Let $u(x)$ be given at the two end-points a and b and at the set of n midpoints $t_i = (x_{i-1} + x_i)/2$, $i = 1, \cdots, n$. That is, the abscissae of the data points are a, t_i, $i = 1, \cdots, n$, and b. For convenience, set $t_0 = x_0$ and $t_{n+1} = x_n$. Assume $u(t_i)$, $i = 0, \cdots, n+1$ are given. We seek a quadratic spline approximation $Q(x)$ to $u(x)$, that interpolates $u(x)$ at $(t_i, u(t_i))$, $i = 0, \cdots, n+1$. We write $Q(x)$ as a linear combination of the B_i's, i.e.,

$$Q(x) = \sum_{j=0}^{n+1} c_j B_j(x), \tag{2}$$

where the scalars c_j are unknowns to be determined (called **coefficients** or **degrees of freedom** - DOFs). In order to determine the DOFs, we impose the **interpolation conditions** at the data points, i.e.,

$$Q(t_i) = u(t_i)\ , \text{ for } i = 0, \cdots, n+1. \tag{3}$$

The interpolation conditions form a linear system (the system of **interpolation equations**) of $n+2$ equations with respect to the $n+2$ unknown DOFs. By solving the system, we obtain the DOFs and then the approximation $Q(x)$ can be evaluated at any point in $[a, b]$ using (2).

(a) [*10 points*] Write down the $n+2$ interpolation equations, in matrix form (in the order indicated). Indicate which entries of the matrix are equal to zero. What are the properties (bandedness, symmetry, diagonal dominance) of the matrix of the interpolation equations?

(b) [*5 points*] Based on the definition of the basis functions in (2), obtain a simpler than (2) formula to evaluate the quadratic spline approximation at an arbitrary point in $[a, b]$. Can the formula be simplified even further for a node?

3. [*30 points*] Implement quadratic spline interpolation using MATLAB. Write a MATLAB `function yv = qspline(x, u, xv)` that takes as input the knots in vector `x`, the y-values corresponding to the t-vector abscissae in vector `u`, and the abscissae of the points of evaluation in vector `xv`. The output is vector `yv` which gives the values of the interpolant at the points `xv`. Note that, if $n+1$ is the length of `x`, then $n+2$ is the length of vector `u`, and, if nv is the length of vector `xv`, so is the length of vector `yv`.
Test your program for the following cases of n and for data coming from the following test functions:
$n = 4, 8, 16, 32, 64$
$u(x) = x^2$, $u(x) = x^3$, $u(x) = e^x$ (exp function), $u(x) = \dfrac{1}{1 + 25x^2}$ (Runge function) and $u(x) = \sqrt{1 - x^2}$ (arc function).
Set $a = -1$ and $b = 1$.

Compute and **output** the maximum in absolute value error $\max\limits_{i=1}^{nv} |u(xv_i) - Q(xv_i)|$ of the interpolant at the evaluation points. Compare with the respective error of the cubic spline interpolant $C(x)$ obtained by calling the MATLAB function `yv = spline(x, y, xv)`, where `y` is a vector of the same length as `x` holding the respective y-values. Also compute and **output** the maximum in absolute value errors $\max\limits_{i=0}^{n} |u(x_i) - Q(x_i)|$ and $\max\limits_{i=0}^{n+1} |u(t_i) - C(t_i)|$. For convenience, **table** the results. Do the error results for the test functions $u(x) = x^2$ and $u(x) = x^3$ agree with those expected? Explain.

For each of the test functions exp, Runge and arc, and for each pair of errors for n and $2n$ subintervals corresponding to each interpolant, also compute and output the experimentally observed orders of convergence. Do the convergence results agree with those expected? Explain or comment.

For each of the test functions exp, Runge and arc,
for $n = 16$, **plot** the quadratic and cubic spline interpolants and the function u at the points of evaluation, versus x (xv), in one subplot.
In another subplot, again for $n = 16$, **plot** the error of the two interpolants at the points of evaluation, versus x (xv).
In a third subplot, again for $n = 16$, **plot** the error of the quadratic pp interpolant at the knots, versus x, and the error of the cubic pp interpolant at the knots, versus x (t).
In a fourth subplot, after the loop of n has finished, **plot** the maximum in absolute value error of the two interpolants at the points indicated above, versus n, in log-log scale.

Notes: Use sparse matrices and sparse MATLAB functions (e.g. `spdiags`) whenever a matrix (two-dimensional array) needs to be constructed and/or stored. Use the \ division operator for solving the linear systems in MATLAB. Use exponential format for outputting the error. A script that runs the experiments required is found in file `testQ.m` on the course web page. You should not alter the output format. Uncomment/comment the print statements whenever necessary, and/or modify the code accordingly. You would need to write the `qspline` function (which, in turn, may call a function that evaluates quadratic spline basis functions), and the Runge and arc functions. Hand in a hard-copy of your source code, your plots, output (5 tables), discussion and observations. For this question, you should have a total of 12 plots in 3 pages. Note that the second arguments in `yv = qspline(x, u, xv)` and `yv = spline(x, y, xv)` are both vectors of values of u, but at different points, and of different lengths. Also note that, plotting quantity f versus g means f is in the vertical axis and g in the horizontal one.

4. [*15 points*] 4-point Lobatto (Gauss-Lobatto) rule

(a) Determine w_0, w_1 and α, so that the quadrature rule of the form

$$\int_{-1}^{1} f(x)dx \approx w_0 f(-1) + w_1 f(-\alpha) + w_1 f(\alpha) + w_0 f(1)$$

has the maximum polynomial degree. (Note: The degree of this rule may be higher than the number of parameters to be determined indicates.)

(b) Use transformation of variables in order to transform the rule constructed in (a) into one which is appropriate for approximating $\int_a^b g(t)dt$ in a general interval (a, b). Show the transformation you have used.

(c) What is the polynomial degree (degree of accuracy) of the rule constructed in (b)? Explain.

(d) Based on the rule constructed in (b) obtain a composite (compound) quadrature rule for approximating $\int_a^b f(x)dx$ in a general interval (a, b) with s panels.

(e) Which rule has higher polynomial degree, the 3-point Gauss rule or the rule constructed in (b)? Explain.
Which rule is preferable to use in a composite form and why?
Which rule is preferable to use in an adaptive form and why?

Note: This question is to be done by hand calculations.

5.

(a) [*7 points*] Calculate the exact value of $I = \int_0^1 f(x)dx$ for $f(x) = x^i$, $i = 3, 4, 5, 6$, and for $f(x) = x^2 e^x$.

Approximate $I = \int_0^1 f(x)dx$ using the quadrature rules
(α) Simpson’s
(β) Gauss based on 2 points
(γ) Gauss-Lobatto based on 4 points
once with 1 (sub)interval, and once more with 2 subintervals (panels).

Compute the error (including its sign) for each of the quadrature rules, with 1 and 2 panels, and each function. Comment on the magnitude and the sign of the error of the rules (α) and (β).
Though you can do this question will hand calculations, it is easier and less error-prone to code the rules and do the calculations in MATLAB.

(b) [*8 points*] For the rule (γ), suppose you are not given a precise error formula, but you are given the fact that the error for this rule follows the formula $I - Q_{(\gamma)} = Kf^{(p)}(\eta)(b-a)^{p+1}$, in which you don’t know what K and p are. Using appropriate numerical experiments, make an (educated) guess about what the values of K and p are. Explain. (In this question you are asked to give some justification for the values of K and p you guessed, based primarily on numerical experiments, not to derive the error formula for the rule.)

6. [*10 points*] Consider the function $f(x) = (1-x)(\mathrm{atan}(\nu * (x-\mu)) + \mathrm{atan}(\nu\mu))$ in $[0,1]$. This function has a peak close to $x = \mu$, the sharpness of which is controlled by ν. We would like to test the adaptive quadrature functions `quad` and `quadl` of MATLAB on $f(x)$.
Write a script that, with $\mu = 1/2$, plots $f(x)$ with $\nu = 1, 10, 100$ in one plot. (Use solid, dashed and dotted for $\nu = 1, 10, 100$ respectively.) Then the script should do the following: For all combinations of $tol = 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$ and $\nu = 1, 10, 100$, calls `quad` and `quadl` and outputs the number of function evaluations required and the (actual) error. (For the error, you must use exponential format.) Formulate your output as a table. Comment on the efficiency of the functions. The exact integrals are

```
q = [-3/4*atan(1/2)+1/2 6/25*atan(5)+1/20 2499/10000*atan(50)+1/200];
```

for $\nu = 1, 10, 100$, respectively.