MATH 440A/540A Parallel Scientific Computing
Homework – Exercise Sheet 3

(Throughout, replace `xxxx` with F90+ or C or C++.)

1. Write a `xxxx` procedure to generate a data set with entries $(x_i, y_i),\ \ i = 0, \cdots, dsize - 1$ without and with noise in the range `[-noise_range, noise_range]` (so that `noise_range = 0` corresponds to without noise).

   The dummy list of input values for the subroutine are:
   `a, incr, dsize, func, noise_range`, where `func` is an external function and other values are scalars. The output from the procedure should the data set with

   $$x_i = a + i * incr, \qquad y_i = func(x_i) + rand\_noise, \qquad i = 0, \ldots, dsize - 1$$

   (in vectorized form). where $rand\_noise = 0$ if `noise_range = 0` otherwise $rand\_noise$ is a random number in the range `[-noise_range, noise_range]`.
   (Use the intrinsic subroutine `RANDOM_NUMBER` to generate $rand\_noise$.)

2. Write `xxxx` programs to generate and write four data sets using the procedure in Q. 1, and actual fixed input parameters

   $$aa = 0, \quad incre = 0.1, \quad data\_size = 51, \quad f(x) = x^5 + x^4 + 3x^3 - 4x^2 + 2x + 3$$

   and four noise range `[-noise_par, noise_par]`. with parameter parameter `noise_par =` (i) 0, (ii) 0.1, (iii) 0.5, (iv) 1.0.

3. Write a `xxxx` procedure to fit a data set $(datax_i, datay_i),\ \ i = 1, \cdots, dsize$ (such as those in Q. 2 or any set generated using Q.1) with a general degree n polynomial least square fit defined by

   $$poly\_fit_n(x) = \sum_{i=0}^{n} c_i x^i, \qquad \text{if} \quad 0 < n < dsize. \tag{1}$$

   The unknown $n + 1$ coefficients in (1) are solutions of the $n + 1$-dimensional linear system

   $$\sum_{j=0}^{n} a_{i,j} c_j = \sum_{k=1}^{dsize} (datax_k)^i datay_k, \qquad i = 0, \cdots, n, \tag{2}$$

   where the known matrix entries of the system are defined by

   $$a_{i,j} = \sum_{m=1}^{dsize} (datax_m)^{i+j}, \quad i, j = 0, \cdots, n.$$

   Use your own or LAPACK (or other library) routine to solve the linear system (2).

   For the procedure, use only the following dummy list of parameters:
   `x, y, lea_sq_fit_c, error` so that

   $$(x_i, y_i) = (datax_i, datay_i),\ i = 1, 2, \cdots \qquad (lea\_sq\_fit\_c)_j = c_j, \quad j = 1, 2, \cdots$$

   and the variable `error` should return from the subroutine to any calling program unit with appropriate error message if the inequality condition in (1) is violated.

4. Write `xxxx` programs that use procedures in Q.1 and Q.3, four different input parameter sets in Q. 2 (with and without noise) and the least squares fit polynomial degree $n = 1, 2, 3, 4, 5$ to:
   - Generate several plots (at least one for each input parameter set) of the least squares fit and compare with the actual function that generated the data. For plots, use at least 1000 equally spaced points for every unit interval.
   - Tabulate (with at least one row for each input parameter set) approximate maximum interpolation errors (that is, errors within the domain of x-data values) in fitting data with various input parameters (with appropriate number of columns and headings), where interpolation error is defined using at least $N \geq 10,000$ equally spaced points $u_i,\ i = 1, \cdots, N$, in $[datax_1, datax_{dsize}]$:

$$Int\_Err_n = \max\{|f(u_i) - poly\_fit_n(u_i)| : i = 1, \cdots, N\}.$$

   - Tabulate (with at least one row for each input parameter set) approximate maximum extrapolation errors (that is, errors outside the domain of x-data values) in fitting data with various parameters (with appropriate number of columns and headings), where error is defined using at least $M \geq 1,000$ equally spaced points in $w_i,\ i = 1, \cdots, M$ in $[datax_1 - 1, datax_1]$ and $w_i,\ i = M + 1, \cdots, 2M$ $[datax_{dsize}, datax_{dsize} + 1]$:

$$Ext\_Err_n = \max\{|f(w_i) - poly\_fit_n(w_i)| : i = 1, \cdots, 2M\}.$$

   - Based on the generated figures and tables, write a report to analyze (i) the quality of the fit change as the amount of noise in the data is changed and (ii) the quality of the higher-order fit compared to the lower-order fit for each chosen noise.

5. Suppose that a numerical approximation, to some quantity B (such as definite integral of a function), is denoted by B_h, where h is the only (step-size) parameter that determines B_h using some algorithm. Suppose that a theoretical investigation of the algorism leads to some (unknown) constants $C \neq 0$, $r > 0$, and $\delta > 0$ and the following relation between B_h and B

$$B_h = B + Ch^r + O(h^{r+\delta}) \quad \text{as } k \to \infty,$$

   so that B_h converges to B as $h \to 0$ with rate of convergence r. A sequence approximations to r can be computed without knowing the exact value B.

   This can be achieved by choosing various step sizes h_1, h_2, h_3, ... satisfying

$$h_k = \frac{h_{k-1}}{2} \quad \text{for } k \geq 2. \tag{3}$$

   and computing the resulting approximation $B_k = B_{h_k}$ and approx. rate of convergence

$$r_k = \log\left(\frac{B_{k-1} - B_{k-2}}{B_k - B_{k-1}}\right) \Big/ \log 2 \quad \text{for } k \geq 3,$$

   Theoretically, it can be shown that $r_k \to r \quad \text{as } k \to \infty$. Write a `xxxx` procedure to compute the approximate rate of convergence with input vector parameter (of any size) `B` consisting of various B_k. Output from the procedure should return a vector `r` consisting of the `r_k` values and a message warning the user that the step size assumption in (3) should have been satisfied by the input data. (Note that the length of the vector `r` is two less than that of the input parameter `B`. You may choose to fill this gap in `r` by using two dummy values such as `NaN`.)

6. Write a `xxxx` procedure to compute an approximation to $f'(a)$, the derivative of an input function f at any input point a and a step-size h using the formula

$$D_h = \frac{f(a+h) - f(a-h)}{2h}.$$

It is known that for sufficiently smooth functions, the rate for this approximation is $r = 2$.

7. In order to test the procedures in Q.5 and Q.6, consider (i) any smooth function of your choice; (ii) a point at which you would like to evaluate its derivative; (iii) initial step size h_1; and (iv) the total number N of approximation $D_k = D_{h_k}, \quad k = 1, \cdots, N$ (For example $f(x) = \arctan x, a = 1, h_1 = 10^{-2}, N = 10$.) For your choice, write an `xxxx` program using procedures in Q.6 and Q.7 to print a table (in a three column format with appropriate headings) N step sizes, N numerical approximations, and $N - 2$ apparent convergence rates. Compare your result in the third column with the theoretical rate given in Q. 6.

8. The quantity B specified in Q. 5 for this question is $\int_a^b f(x)\,dx$, for some fixed values a, b and function $f : [a, b] \to \mathbf{R}$. We define three different well known approximations B_h to this quantity, denoted respectively by M_h, T_h, S_h, where $h = \max_{1 \le j \le n} len_j$, by subdividing $[a, b]$ into n subintervals of each of length $len_j, \quad j = 1, \cdots, N$ (need not be equal):

Let $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ be a partition of $[a, b]$ and put

$$len_j = x_j - x_{j-1} \quad \text{and} \quad \xi_j = \frac{x_{j-1} + x_j}{2} \quad \text{for } 1 \le j \le n.$$

Given a function $f : [a, b] \to R$, we write $h = \max_{1 \le j \le n} h_j$ and define

$$M_h = \sum_{j=1}^{n} f(\xi_j) len_j \quad \text{and} \quad T_h = \sum_{j=1}^{n} \frac{1}{2}[f(x_{j-1}) + f(x_j)] len_j, \quad S_h = \frac{2}{3} M_h + \frac{1}{3} T_h,$$

the midpoint, trapezoidal and Simpson rule approximations to $\int_a^b f(x)\,dx$. The data points $x_i, i = 0, \cdots, n$ used for M_h, T_h, S_h are called quadrature points.

Theoretically, for sufficiently smooth non-periodic integrands, the rate of convergence of M_h and T_h is 2 and that of S_h is 4. However, for smooth periodic functions, with some of the derivatives of the integrand periodic on $[a, b]$, the trapezoidal rule T_h achieves faster rate of convergence provided the quadrature points are equally spaced.

Write a `xxxx` procedure with dummy input variables being an external integrand function and a vector of quadrature points and the number of number of output variables is up to the user: All of M_h, T_h, S_h or either one or two of these approximations.

9. Write a `xxxx` program using procedures in Q.8 and Q. 5 to compute approximate values of the integral $\int_a^b f(x)\,dx$, for any of your choice of a, b and f using the mid-point, trapezoidal and Simpson rule and the associated apparent rates of convergence, using $n+1$ equally spaced quadrature points on $[a, b]$ with $n = 2^k, k = 1, 2, \cdots, 10$. Tabulate values as specified in Q. 7 for T_h, M_h, S_h and compare your results with theoretical rates of convergence described in Q. 8.

(For example, you may consider $a = 0, b = \pi$ and $f(x) = \exp(\sin(x) + \cos(x))$ for non-periodic case and $f(x) = \sin(x) + \cos(x) + x^2/\pi$ for periodic case test.)