Part 1: Polynomial Interpolation

After reviewing the Matlab Notes on pp. 125–139 of the textbook, do the following exercises:

- By hand: Problems 4.3.3, 4.3.4, 4.3.7 on page 119.
- In Matlab: Problems 4.5.3, 4.5.4, 4.5.6 on page 131; 4.5.13, 4.5.14 and 4.5.15 on page 139.

Part 2: Least Squares Approximation

Write a Matlab function that constructs and plots the best least squares $m$th degree polynomial fit to an arbitrary data set $\{(x_i, y_i)\}_{i=0}^n$, with $m \leq n$ and $n \geq 1$. The function should take the vectors $x = (x_i), y = (y_i) \in \mathbb{R}^{n+1}$ and the nonnegative integer $m$ as input, and return the vector $a = (a_k) \in \mathbb{R}^{m+1}$ of coefficients of the least squares polynomial $p_m(x) = \sum_{k=0}^m a_k x^k$, as well as a plot of the graph of $p_m(x)$ on the interval $[x_0, x_n]$. The plot should clearly display the data points $\{(x_i, y_i)\}_{i=0}^n$, denoted for example as small circles.

In addition, the function should return the least squares error

$$\varepsilon_{LS} = \left[ \sum_{i=0}^n \left( y_i - \sum_{k=0}^m a_k x_i^k \right)^2 \right]^{\frac{1}{2}}$$

(this can be easily computed as the 2-norm of the residual vector $r = y - Aa$, where $A \in \mathbb{R}^{(n+1)\times (m+1)}$ is the Vandermonde-type matrix constructed from the vector $x$).

You should verify that when $n = m$, the computed approximation polynomial is actually the interpolation polynomial (so that $\varepsilon_{LS} = 0$ in this case). The code should also work in the case $m = 0$, in which case $p_0(x) = a_0(= \text{const.})$ should be simply the arithmetic average of the $y_i$ values.

You should make sure that the code runs as efficiently as possible, and that the graphics is of good quality. For instance, Horner’s nested multiplication scheme should be used when evaluating the polynomial $p_m(x)$ at a given point, and the polynomial should be evaluated at sufficiently many points when the plot is constructed (so that the graph looks nice and smooth).

Also, the code should contain checks and provide error messages (for example if $m > n$, or $n < 1$ or $x_i = x_j$ for some $i \neq j$). You should test this feature by intentionally calling the function with inadmissible parameter choices.

Use your code to produce least squares polynomial approximations of degree $m = 0, 1, 2, 3, 4, 5$ to the following data sets:

1. The points $(-5, 10), (-4, 5), (-3, 1), (-1, 0), (0, 0), (1, -3), (2, 4), (4, 1), (4.5, 3), (5, -1)$.

2. The values of the function $f(x) = e^{-x^2} \sin x$ at 11 equally spaced points on the interval $[-\pi, \pi]$.

As before, produce plots of the approximating polynomials and data points. For the data set in item 2 above, include the graph of $y = f(x)$ in your plots. Be sure to produce nice plots where all plotted quantities are clearly labeled and recognizable.