Due Date: Monday, 14 January, 2019

Read before starting: download the Matlab files *hw00Bpractice.m* and *hw00Bpractice.mat* from the Canvas assessments table. Complete this worksheet on a computer version of Matlab (laptop, or lab computer). Only after you have completely understood the coding of the best-fit quadratic should you attempt the *math-apps* online version. You may submit more than once, but only your last score is the one saved by the Canvas gradesheet. Note, there is a daily limit to the number of submissions you can send to the server — you have now been warned. Also, if you are new to Matlab, do NOT leave the submission to the last day!

Begin by completing this worksheet. Answer all of the questions below in the blank spaces. After you have done this correctly, you will be ready to proceed to the online version. If you encounter difficulties, bring your questions and scripts to the Computing Workshop (to be announced in lectures).

B) Linear Algebra

The following problem will review basic linear algebra operations in Matlab. The code you have been provided performs a linear least squares fit on some noisy data. We will modify it so that it instead performs a quadratic least-squares fit on the same data. You do not need to know the best-fit theory, the worksheet below is designed to be a warm-up to matrix-vector arithmetic in Matlab.

- Start by running the code *hw00Bpractice.m* as is. There is information in the screen output and the plot. The code starts by loading the practice data from the file *hw00Bpractice.mat*; and the *whos* command gives a list of the data variables. List below all variables and identify them as scalar, vector or matrix quantities (with sizes):

- The random dataset you begin with are N coordinates: $\mathcal{S}_N = \{(t_j, y_j) \mid j = 1 \ldots N\}$. These are the blue circles in the plot. The best-fit linear function is shown as the red line — give, to 3 significant digits, the linear function $z(t)$ that is plotted.

- The mysterious data variables M and v are a matrix and a vector — these somehow generate the best-fit linear function by the linear solve: $[M]\,\vec{c} = \vec{v}$ — give, to 3 significant digits, the linear algebraic system that is being solved.

- Begin by modifying the code so that you are doing the calculation of the M and v variables yourself. An intermediate matrix has been defined for you — A is an $N \times 2$ matrix whose first column equals t and whose second column equals all 1's. After redefining $[M] = [A]^T [A]$ and $\vec{v} = [A]^T\,\vec{y}$ — again give the linear algebraic system that is being solved.

- Verify that your plot looks as it did originally, and compute the average squared-error:

$$Err = \frac{1}{N}\sum_{j=1}^{N}(y_j - z(t_j))^2 = \_\_\_\_\_ .$$

  Note that if you change the c_j coefficients from the best-fit values, this error must be larger!
- Now you are ready to perform the quadratic fit, but note that you will have to be aware of Matlab's element-wise arithmetic. Modify A to be $N \times 3$, with the first column equalling t^2, the second column equalling t, and the third column equalling all 1's. Give the new values of M and v.

- Finally, modify the Matlab vector z so that $z_j = c_1 t_j^2 + c_2 t_j + c_3$, where c_j are the entries of $\vec{c}$ — give the best-fit quadratic polynomial.

- Finally, check that your plotted quadratic looks like the best-fit polynomial, and calculate the average squared-error again:

$$Err = \frac{1}{N}\sum_{j=1}^{N}(y_j - z(t_j))^2 = \_\_\_\_\_ .$$

  What do you notice about the value of this error?

- You are now ready to do the online version of this warm-up. Note that the random data will be different, but the best-fit procedure is the same as in this worksheet.
- Note, the graded variables are: *z*, *Err*.
