**Instructions**: Present your work in a neat and organized manner. Please **use** either the $8.5 \times 11$ size paper or the filler paper with pre-punched holes. Please do **not use** paper which has been torn from a spiral notebook. Please secure all your papers by using either a staple or a paper clip, but **not** by folding its (upper left) corner.

You **must** show **all of the essential details** of your work to get full credit. If you used *Mathematica* for some of your calculations, attach a printout showing your commands and their output. If I am forced to fill in gaps in your solution by using notrivial (at my discretion) steps, I will also be forced to reduce your score.

Please refer to the syllabus for the instructions on working on homework assignments with other students and on submitting **your own** work.

# Homework Assignment # 10

**In Problems 1, 3, and 5 you will be asked to run a Matlab code that I posted online and also to type some Matlab commands outside of this code. If you need help understanding the meaning or the syntax of a particular command** `commandname`**, you may either type** `help commandname` **(e.g.,** `help rand`**) or look inside the code to see how I used that command. If you are not sure what the command name may be, click on Matlab's *Help* tab (in the right corner of the menu bar at the top of the command window) and type in some related key word, e.g.,** `round`**, into the search box. Finally, if nothing else works, asking Google should help in most cases; e.g., try it with "triangular matrix in matlab" (omit the quotes, of course).**

1. Download the code `hw10_p1.m`.

   (a) Taking `ep=0.001`, compute $h_{\rm opt}$ from Eq. (10a) while taking $M = \max\limits_{0\le t\le t_{\max}} \left(\frac{1}{2}|u''(t)|\right)$, where $u$ is the unperturbed solution:

   $$u = \ln(t+5) + 0.4t\,.$$

   Compute (by hand) the corresponding $n_{\rm opt} = t_{\max}/h_{\rm opt}$ and round it to the nearest integer.

   Now, in `hw10_p1.m`, set $n$ to equal to each of the values from the list

   $$n \;=\; \{0.25,\; 0.5,\; 0.75,\; 1,\; 1.5,\; 2,\; 3,\; 4\}\cdot n_{\rm opt}$$

   (of course, rounding to the nearest integer whenever necessary) and run the code for one $n$ at a time. Record the average of the absolute value of the error in the numerically reconstructed $r_{\rm num}(t)$ for each given $n$. (The exact data $r_{\rm exact}(t) \equiv u'(t)$ is computed in the code. Matlab's command for finding the average of the absolute difference between any two vectors $f$ and $g$ is `mean(abs(f-g))`.)

   Plot these error values versus $n$ in a separate plot. Comment on whether your plot qualitatively confirms the behavior predicted by Eq. (9) of the Notes.

   Attach plots of $r_{\rm num}(t)$ for $n = n_{\rm opt}$ and one other value of $n$ of your choice, so as to illustrate the importance (or otherwise) of choosing a nearly optimal value for $n$ (and hence for $h$).

   (b) Repeat the steps in part (a) for `ep=0.05`. Compare the values of `mean`$(|r_{\rm num}(t) - r_{\rm exact}(t)|)$ for $n = n_{\rm opt}$ in parts (a) and (b). Does this comparison confirm the square-root dependence of the error on the noise amplitude exhibited by the r.h.s. of Eq. (10b)?

   *Hint*: Recall that if a quantity $a = O(\epsilon^k)$ for some $k$, this means that $a \approx C\cdot\epsilon^k$ for some unknown constant $C$. Then, if $a_1$ and $a_2$ are the values of $a$ obtained for $\epsilon = \epsilon_1$ and $\epsilon = \epsilon_2$, respectively, how are the ratios $(a_1/a_2)$ and $(\epsilon_1/\epsilon_2)$ related to each other?

2. Repeat the analysis of Secs. 10.1 and 10.2 where the finite-difference approximation (3) is now replaced with

$$u'(t) \approx \frac{u(t+h) - u(t-h)}{2h} \,.$$

To obtain the error made by this approximation in the smooth part of the derivative $u'(t)$, use one more order in the Taylor expansion than in the notes. To estimate the contribution to the error from the noise, use the worst-case argument similar to that used in the notes.[1] Obtain expressions for $h_{\rm opt}$ and the error for $h = h_{\rm opt}$.

To verify your answer, find and quote a formula in Lecture 10 [2] that shows the dependence of the optimal error on $\epsilon$.

3. This problem is worth **1.5 points**.
In this problem, you will confirm that matrix $A$ defined in Sec. 10.4.2 smoothens any fast ripple or discontinuities presented in the data $w(x)$ (see Eqs. (23a) and (21), (22) of the Notes). This will be explored in parts (a) and (b). As a consequence, reconstruction of the data from a solution $\mu(x)$ (see (23)) becomes essentially impossible, because a tiniest ripple in the solution will "cause" (in the sense defined at the beginning of Lecture 10) a humongous ripple in the data. This will be explored in part (c).

(a) At the end of Sec. 10.4 we presented a calculation to the effect that even a discontinuous gold density $w(x)$ will result in a smooth disturbance of the vertical force $\mu(x)$. Here you will show this by explicitly calculating the integral in (21) for a particular discontinuous $w(x)$. For simplicity, set $\gamma = D = 1$ and $s_{\rm min} = 0$ and $s_{\rm max} = 1$ in the integral in Eq. (21).

Consider a very simple discontinuous function

$$w(x) = \begin{cases} 0, & 0 \le x < 0.5; \\ 1, & 0.5 \le x \le 1 \,. \end{cases}$$

Use Mathematica to compute $\mu(x)$ given by Eq. (21), and also $\mu'(x)$ and $\mu''(x)$ for $-1 \le x \le 2$. See Hints and Notes below.

Make plots of $\mu(x)$, $\mu'(x)$, and $\mu''(x)$ and attach their printouts to your work. State whether you see a discontinuity for any values of $x$ in any of these plots.

*Hint 1*: Although it is possible to define $w(x)$ using the command `Piecewise` in Mathematica, there is an easier and less time-consuming method to compute the required integral.

*Note 1*: When you define $w$ in Mathematica, it does not matter which letter you use to denote its argument (as long as you do denote one). So, it can be $x$, or $x = s$, or anything else. When you compute the integral, however, the argument of $w$ must be named the same as your integration variable (I recommend that you use $s$ as in the Lecture notes to avoid any confusion).

*Note 2*: You probably have noticed that the range of the argument of $\mu$ is greater (wider) than that for $w$. This is because the two ranges are not related to one another. The argument of $w$ ($s$ in Lecture notes) pertains to the locations where gold is found. The argument of $\mu$ ($x$ in Lecture notes) pertains to the locations where measurements are taken.

(b) The goal of this part is to obtain a result equivalent to that in part (a), but in the *discrete* rather than continuous setting. That is, instead of exploring the "smoothing power" of the integral in Eq. (21), you will explore the "smoothing power" of matrix $A$ in Eq. (23a).

[1] The worst-case noise will be *different* from that considered in Secs. 10.1 and 10.3.
[2] It is *not* displayed conspicously.

Download the code `hw10_p3.m` and run it. This will set up a matrix $A$, defined in Sec. 10.4.2, and a *column* vector `xx` of length $n = 20$. This `xx` is the counterpart of the discrete vector $\{x_i\}$ defined before Eq. (22).

We will now compare how matrix $A$ transforms two vectors. One will correspond to a jagged vector $\underline{w}$ (defined after (23a)); here it will be called $\underline{\mathbf{y}}$. The other will be a constant (i.e., smooth) vector having the same average (across its entries); this smooth vector will be called $\underline{\mathbf{Y}}$ here. Thus:

$$\underline{\mathbf{y}} \equiv \underline{w}_{\text{jagged}}, \qquad \underline{\mathbf{Y}} \equiv \underline{w}_{\text{smooth}}\,.$$

To do so, begin by re-reading the boldfaced preamble to this Homework Assignment (the entire Assignment, not just Problem 3). Then, at the prompt in the command window, define a random column-vector $\underline{\mathbf{y}}$ of length $n$ such that the nominal[3] average of this vector is to equal 1 and the nominal maximum deviation from the average is to also equal 1. (If $\underline{\mathbf{y}}$ had been a scalar rather than a vector, you would type `y=2*rand-1`. You need to modify this for a column-vector of length $n$.) This is your jagged vector.

Next, define another vector using the command `Y=mean(y)*ones(size(y)).` `Plot` both vectors versus the variable `xx` in the same figure using different line styles.

Now open a new figure by typing, say, `figure(2)`. Compute and then, in figure 2, plot together "the solution" vectors $\underline{\mathbf{v}} = A\underline{\mathbf{y}}$ and $\underline{\mathbf{V}} = A\underline{\mathbf{Y}}$ versus `xx`. (Recall that: $\underline{\mathbf{v}}$ is obtained from the jagged vector $\underline{\mathbf{y}}$, and $\underline{\mathbf{V}}$ is obtained from the smooth vector $\underline{\mathbf{Y}}$ that has the same mean value as $\underline{\mathbf{y}}$.) Use the same line styles for these two graphs as in the previous figure. This plot that you have just created must be the first `subplot` out of three that will be plotted in this figure. Label the vertical axis of your subplot as `'v and V'` (see `help ylabel`).

In the second and third subplots, plot the numerically computed first and second derivatives of your "solutions" $\underline{\mathbf{v}}$ and $\underline{\mathbf{V}}$. Specifically, in the second subplot, plot `diff(v)./diff(xx)` and `diff(V)./diff(xx)` versus `xx(1:end-1)`. In the third subplot, plot `diff(v,2)./(diff(xx(1:end-1))).^2` and `diff(V,2)./(diff(xx(1:end-1)).^2)` versus `xx(1:end-2)`. Use the same line styles as before and also label your vertical axes. Print out both of your figures.

Draw a *common* conclusion from your results in parts (a) and (b). (Re-read the preamble to Problem 3 and that to its part (b) (i.e., this part) if you are not sure what kind of conclusion you are supposed to draw.)

(c) Clear the workspace from the previous part. Inside the code, set `epsilon=0.0001` and then run the code. Examine the figures, and *especially note the vertical scale of figure 11 !*
Next, type `figure(11); hold on` in the command window. This will prevent the next plot that will be made in this figure from erasing the graph that is already there. Do the same for figure 13. *Without clearing the workspace*, run the code again. (Since you did not clear the workspace, the random vector added to $\mu$ will be different this time.) Repeat this four more times. Print out figures 11 and 13.

Draw a conclusion from this part of the exercise.

Food for thought, to be digested in Lecture 11 In figure 13 you see the profiles of the perturbations that you have added to $\mu(x)$ and which generated the jagged curves in figure 11. Do the profiles in figure 13 seem to have similar shapes? Can one say the same about the profiles in figure 11?

4. Illustrate the derivation of the condition number in Sec. 10.5 geometrically, using as an example

$$A = \begin{pmatrix} 0.01 & 0 \\ 0 & 100 \end{pmatrix}.$$

[3] Average values quoted in Matlab's description of its command are nominal in the sense that their values are guaranteed only in the limit of very long such vectors or very many realizations of the same command obtained with different seeds of the random number generator. See the Bonus problem below for some more details.

Specifically, do the following. Divide the page into two halves with a vertical line. On one side draw vectors $\underline{b}$ and $A^{-1}\underline{b}$. On the other side, draw $\underline{b} + \underline{\Delta b}$ and $A^{-1}\underline{b} + \underline{\Delta b}$. Here $\underline{b}$ and $\underline{\Delta b}$ are the special vectors introduced before Eq. (28). Add a brief explanation to your drawings (i.e., how they illustrate the concept of the condition number).

5. This problem is worth **1.5 points**.

(a) Run the code `hw10_p3.m` with $n = 5,\ 10,\ 20,\ 30$, and $40$ and each time record the condition number of $A$. Then plot cond($A$) as a function of $n$. You may need to use a logarithmic scale for the vertical axis to show your results in a legible manner. Attach the printed plot.

*Note*: Matlab's command for the logarithm to base 10 is `log10`.

Why do the results for $n = 30$ and $n = 40$ not follow the trend of the results for the smaller $n$?
*Hint*: What size is Matlab's round-off error? (If you are not sure, type `eps` at Matlab's command prompt.)

(b) For $n = 10$, plot all the eigenvalues of $A$. (The corresponding Matlab's command is `eig`.) Again, you may need to use a logarithmic scale for the vertical axis. Attach the printed plot. Also, verify that

$$\mathrm{cond}(A) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}\,.$$

*Note*: To plot a vector `myvector` versus its index, simply type `plot(myvector)`.

(c) Write a short code where you can set up the matrix in Eq. (33) (not including the factor $1/h$) with any dimension $n$. Run this code for $n = 10,\ 20,\ 50,\ 100,\ 200,\ 300,\ 500$ and record the condition numbers. Plot cond($A$) as a function of $n$ (you will *not* need a logarithmic scale in this case.) Attach the printed plot along with a printout of your code.

*Technical note*: For setting up the matrix of Eq. (33), you may use a number of methods. For example, you may use two `for`-loops; the corresponding syntax is explained in the Matlab Primer posted online. Alternatively, you may once again reread the preamble to this Assignment.

(d) Compare the rates with which the condition number grows with $n$ in parts (a) and (c).
Of the problems of inverting the matrices referred to in parts (a) and (c), which one will be more ill-conditioned? What does this mean in terms of the error when solving $A\underline{\mathbf{x}} = \underline{\mathbf{b}}$?

Bonus (worth **0.5 pt**; credit will be given only if the solution is mostly correct)

If you have not been satisfied with a result in Problem 1(b), do the following. For each `ep`, immerse the code of `hw10_p1.m` in a `for`-loop which will record values of $\mathtt{mean}(|r_{\mathrm{num}}(t) - r_{\mathrm{exact}}(t)|)$ for each random realization and at the end will take their average over the number $N$ of all realizations.[4] Of course, you should save your code under a name different from `hw10_p1.m`.

Have your conclusion about the the dependence of $\mathtt{mean}(|r_{\mathrm{num}}(t) - r_{\mathrm{exact}}(t)|)$ on $\epsilon$ following formula (10b) improved compared to that in Problem 1(b)?

[4] By default, Matlab changes the seed of the random number generator in each realization inside the for-loop. However, each time that you run your code, you will still get (slightly) different answers. This is because the initial seed in the for-loop is chosen by Matlab according to some internal algorithm. To obtain results that do not change from one run to another, specify this initial seed by a command rand('state', any integer number, e.g., 0).