18.085, PROBLEM SET 3, DUE 7/7, 9:30AM IN CLASS

Question 1. (25 pts.) Resonance and practice with (complex) exponentials. In this question, to be done entirely by hand, $i = \sqrt{-1}$, the unit imaginary number. Recall that the equation for one mass (of mass m) and one spring (of spring constant c), where u is the position of the mass away from equilibrium, is (no forcing):

$$mu'' + cu = 0 \tag{0.1}$$

where the prime $'$ means a derivative in time, and $u = u(t)$ is a function of time only (no space). We omit initial conditions here.

a) What is the natural frequency of this spring and mass system, that is, the number ω such that both $u_1 = \sin \omega t$ and $u_2 = \cos \omega t$ are solutions to equation (0.1)?

b) Circle your answers:

The mass oscillates FASTER or SLOWER when it is heavier.

The mass oscillates FASTER or SLOWER when the spring is stiffer (larger c).

c) Since complex exponentials are basically like sines and cosines (recall $e^{it} = \cos t + i \sin t$), we can guess instead that the solution will look like $u = e^{rt}$, where r is some number to be found, possibly complex. Plug in the guess $u = e^{rt}$ into equation (0.1) to find the two possible values of r. That is, find r_1 and r_2 such that $u_1 = e^{r_1 t}$ and $u_2 = e^{r_2 t}$ are solutions to (0.1).

d) Write r_1 and r_2 from c) in terms of ω, the natural frequency in a).

e) BONUS QUESTION! 5 pts. More realistic: add damping to the mass! We are now looking at the equation

$$mv'' + bv' + cv = 0, \tag{0.2}$$

where b is some damping coefficient (say coming from friction). Because of the damping, the amplitude of the oscillations of the mass should decrease (we are losing energy). Try again a guess of $v = e^{st}$, now in equation (0.2), and find the two values of s such that the guess is a solution of (0.2). That is, find solutions $v_1 = e^{s_1 t}$ and $v_2 = e^{s_2 t}$ that satisfy equation (0.2).

f) BONUS QUESTION! 5 pts. Do your solutions in e) decay in time as expected when damping is added? You should find two different cases.

g) Resonance! We are now looking at the equation (no damping, but with forcing)

$$mw'' + cw = f, \tag{0.3}$$

where $f = f(t)$ is some exterior forcing. Assume $f(t) = e^{r_1 t}$, with the r_1 you found in c). (You could use r_2 instead to get the same result, but no need to.) This means we are forcing the system as its natural frequency. Now u_1 and u_2 from c) are not solutions anymore, because f is not 0. Try $w = Ate^{r_1 t}$ as a guess for the solution. What A do you get?

h) What happens to the solution $w = Ate^{r_1 t}$ you found in g) as time goes to infinity?

The phenomenon in g) and h), of forcing the system at its natural frequency, is called resonance. Sometimes it's bad (your mass oscillates wildly until the spring breaks), sometimes it's good (your radio tuner only amplifies the exact frequency corresponding to the station you wish to listen to).

Question 2. (25 pts.) Least squares, the normal equations, and matrix conditioning.

In this question, we will see how bad matrix conditioning can lead to awful errors. Use Matlab. For questions surrounded by `***`, show the Matlab code you used and the result you obtained if appropriate (ok to rewrite by hand).

We want to find the best fit, in the least squares sense, to our data. It turns out that our data fits exactly a degree 12 polynomial (in practice, you would not know that), but the data is noisy as you expect from taking measurements. This polynomial that fits our "true" data (without the observation error) is

$$p(x) = 5 + 4x - 7x^2 + 1x^3 + 0x^4 - 2x^5 + 4.9x^6 - 3x^7 + 0.5x^8 + 1x^9 + 3x^{10} - 2x^{11} + 2x^{12}.$$

Since in practice we would not know this p, we will try to fit our data to a polynomial of arbitrary degree, say 14 (this means that $n = 15$, a nice round number).

We evaluate the data at $m = 100$ equally-spaced intervals in the $[0, 1]$ interval, so that we observe at times $t_j = j * h$ where $h = 1/(m + 1)$, $j = 1, \ldots, m$. This means we have m observations, that we need to fit as closely as possible to a polynomial of degree 14 ($n = 15$).

a) We are now fitting a degree-14 polynomial to our data, instead of a line or a quadratic. `***` Construct the matrix A in Matlab for this least squares problem, and compute it's condition number using `cond(A)`. Do not show me the A you obtained, just show your code plus the result for the condition number. `***`

b) Clearly, if the noise was 0, the solution $\vec{u}$ to the least squares problem would be the coefficients of $p(x)$, the vector $\vec{b}$ would be the polynomial $p(x)$ evaluated at each t_j, and the least squares error would be 0 because $\vec{u}$ is the exact solution. In Matlab:

```
u=[5 4 -7 1 0 -2 4.9 -3 0.5 1 3 -2 2 0 0]';
```

`***` Construct the vector $\vec{b}$ in Matlab, *using the matrix A*. Do not show me the $\vec{b}$ you obtained, just show your code! `***`

c) But the noise is not 0, so our $\tilde{b}$, which is close to $\vec{b}$ but with small observation error, will be, again in Matlab:

```
btilda=b+(10^(-12))*rand(m,1);
```

The function `rand(m,1)` produces an m-vector with each entry independent and uniformly distributed in $[0, 1]$. Since we multiply this random noise by a factor of 10^{-12}, the observation error in $\vec{b}$, that is, the error between $\vec{b}$ and $\tilde{b}$, should be quite small as well. Remember we are interested in fact in the relative error

$$\|\vec{b} - \tilde{b}\| / \|\vec{b}\|, \qquad \|\vec{x}\| = \sqrt{\vec{x}^T \vec{x}}.$$

`***` Compute in Matlab the relative error in $\vec{b}$. Show your code, plus the result you obtain for the error. `***`

d) So we have made a small error in $\vec{b}$. Unfortunately, this error will grow when we solve the normal equations to find the best least squares fit to $\tilde{b}$. This best fit is called $\hat{u}$. (I'm tempted to add a tilda to $\hat{u}$ as well, to remember it is the best fit to the $\tilde{b}$ with the error, but the notation will get too heavy.) This $\hat{u}$ should be close to $\vec{u}$, the true fit when there is no noise. The error we make in the solution is thus

$$\|\vec{u} - \hat{u}\| / \|\vec{u}\|.$$

*** Compute in Matlab the best fit $\hat{u}$ using the normal equations, and compute the relative error in the solution. Show your code for both, plus the result you obtain for the error. ***

e) Fortunately, the fact that the normal equations have bad conditioning is well-known. In fact, Matlab uses a better algorithm, derived from the normal equations but slightly different, which is more stable and leads to less error growth. If you try `uc=A\btilda` in Matlab, it recognizes the fact that A is square, and hence it finds the least squares solution automatically, using this better way. *** Compute in Matlab this new, more clever solution to the least squares problem, and compute the relative error in the solution. Show the result you obtain for the error, *and compare to the error in d).* ***

There are a few lessons to be learned here! First, the least squares matrix for polynomial fitting on an equispaced grid in t has *huge* condition number, which leads to large error growth. Instead, if you *must* do polynomial fitting, use Chebyshev grids (in chapter 4 of book). Or, fit to a polynomial with smaller degree. If we had used a smaller n in this example, the conditioning of A would not have been quite so bad. Second, the normal equations have even worse conditioning than you would think (mainly because we invert A^TA instead of just A), but there are ways around this as we see in e).

Question 3. (25 pts.) Finite differences in time.
In this question, we explore the Forward Euler and Leap-Frog methods, and resonance again. We start with the typical initial conditions $u(0) = 1$ and $u'(0) = 0$. We use $v = u'$ to transform the equation $u'' + u = 0$ into a first order system of equations as we did in class: $u' = v$ and $v' = u'' = -u$. File `pset3_3.m` from the class website (almost) implements Forward Euler on a system of one mass and one spring. It solves the system up to time $t = T = 2\pi$, which amounts to going once around the unit circle in the phase plane. The phase plane is plotted by the file. We also display the error in u and v at the final time T.

a) Change lines 16 and 17 of file `pset3_3.m` so that Forward Euler is used to solve the equation. It's ok to just show me those lines you changed, or you can print the whole file too if you'd like.
b) Try a few different values of `n` (which causes `dt` to change). What behavior do you see in the error at time T as `dt` becomes smaller, and is that what you expect from Forward Euler? (Again, please show a loglog plot as we have done before, and comment on the slope of the line.)
c) Look at the solution plotted in the phase plane, again for a few different `n` (no need to print it though). Describe the behavior of the solution in time, and say whether that is what we expect from our discussion of Forward Euler.
d) Now change file `pset3_3.m` again so that Leap-Frog is used to solve the equation. It's ok to just show me the lines you changed.
e) Same as in b): Try a few different values of `n`. What behavior do you see in the error at time T as `dt` becomes smaller, and is that what you expect from Leap-Frog? (Again, please show a loglog plot, and comment on the slope of the line.)

f) Same as in c): Look at the solution plotted in the phase plane, again for a few different `n` (no need to print it). Describe the behavior of the solution in time, and say whether that is what we expect from our discussion of Leap-Frog.

Now we add forcing, so we solve $u'' + u = f(t)$. This means that, again, $u' = v$, but now $u'' = f(t) - u$.

g) BONUS QUESTION! 5 pts. Let $f(t) = 1/10$, a constant force, say gravity. Modify file `pset3_3.m` to solve to time T using Leap-Frog and $f(t) = 1/10$. Again, show me how you modified the file for this. No need to plot the error. Explain in one sentence how the solution now looks different with a constant forcing term.

h) Now let $f(t) = cos(t)$: we are forcing the system at its natural frequency! Modify file `pset3_3.m` for that new f, *which depends on t*. Again, show me how you modified the file for this. No need to plot the error. Explain in one sentence how the solution now looks different with this special forcing term.

Question 4. (25 pts.) Kirchhoff's Laws.

In this question[1], we will practice solving Kirchhoff's laws on a grid. We consider an $N \times N$ square grid, and number the nodes by rows. We assume that the resistance $R = 1$ along each edge, and that there are no voltage sources. Recall from lecture that the relevant equations are

$$\begin{aligned} \vec{e} &= \vec{b} - A\vec{u} \\ \vec{w} &= C\vec{e} \\ \vec{f} &= A^T\vec{w} \end{aligned}$$

a) For $N = 5$, write down the main diagonal of A^TA. How many zeros does A^TA have? You should not compute A^TA (it's rather big!), but use the symmetry of the problem to justify your answer.

b) In this exercise, you will compute the grid resistance between closest neighbors. Assume that N is odd, and that the node at the center of the grid is grounded. Send a current $f = 1$ into its neighbor to the right. Use `Matlab` to solve for the voltage at this node. Do this for all odd values of N from $N = 3$ to $N = 61$, and produce a plot of the grid resistance as a function of N.

A few useful hints:

i. The exercise asks only for the voltage, so you may want to eliminate $\vec{w}$ from the equations above and write a single equation for $\vec{u}$.

ii. The Laplacian matrix $L = A^TA$ for an $N \times N$ grid may be created by the following `Matlab` code:

```
B = toeplitz([2 -1 zeros(1, N - 2)]);
B(1,1) = 1;
B(N,N) = 1;
L = kron(B,eye(N)) + kron(eye(N),B);
```

iii. Recall that the jth row of a matrix M can be removed using $M(j,:) = [\,]$.

[1]This question is adapted from the exercises on page 155 of the textbook.

iv. Don't forget that grounding a node effectively removes it from A, and changes the numbering of some of the nodes.

c) Use your answer to part (b) to guess the grid resistance between closest neighbors on an infinite grid ($N \to \infty$).

d) Repeat part (b), this time computing the grid resistance between diagonal neighbors. To do this, send a current $f = 1$ into the grounded node's diagonal neighbor to the lower-right, and find the voltage there.

Note: Please attach your code for parts (b) and (d).

Bonus question! (10 pts.) Leap-frog revisited.

In question 3, we use the Leap-Frog method with the first-order system formulation of $u'' + u = 0$, that is, we use Leap-Frog on the equations $u' = v$ and $v' = -u$. Eliminate the v_j's from this Leap-Frog method to obtain the Leap-Frog method that solves directly the equation $u'' + u = 0$:

$$\text{Leap-Frog on } u'' + u = 0 \text{ is} \qquad \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta t^2} = -u_j.$$

(You might recall how the left-hand side approximates u'', and of course the right-hand side has to be $-u$ since $u'' = -u$.)