PX390, Spring 2020, Assignment 5: mechanical crystals

January 14, 2020

A periodic array of solid layers of material, with varying mass densities and elasticities, is being used to try to damp mechanical oscillations (for exampe, to prevent sound transmission). The problem treats all the spatial variation as being in one direction, so it is spatially 1D. Note that this problem is closely related to, for example, x-ray wave scattering from crystals: hence the idea of a mechanical crystal.

The quantity of interest is the displacement $h(x,t)$ from equilibrium as a function of position x in this structure and time t, that results due to an external force, which is sinusoidal in both position and space, $F(x,t) = \exp(i[Kx - \omega t])$ applied to the material. The displacement and force are written using complex variables, and the code will output these complex values: physical interpretation (which is not required for this assignment) requires taking the real parts of these variables.

The unit cell to be solved on is $x \in [0, L)$.

The motion as a result of this time-varying force may be solved via:

$$-\omega^2 \mu(x) h(x) = \frac{\partial}{\partial x}\left(E(x)\frac{\partial h}{\partial x}\right) + \exp(iKx) + i\omega\gamma h(x) \tag{1}$$

where $E(x)$ is the elasticity, $\mu(x)$ is the mass density, and γ is a damping coefficient. We are here only solving the inhomogenous part of the full time-dependent problem, with $h(x,t) = h(x)\exp(-i\omega t)$. Note that what you will be solving for numerically is just the spatially-varying quantity $h(x)$.

Boundary conditions are quasi-periodic. The density and rigidity properties are periodic in the standard sense, with $\mu(x) = \mu(x+L)$ and $E(x) = E(x+L)$. The displacement has the modified periodicity $h(x+L) = \exp(iKL)h(x)$ in order to be compatible with the forcing (in other words, this is a Bloch wave).

You are required to use the LAPACK library to invert matrices, so the compiler flags are different this time. This library is installed on the lab computers and on daisy. Some instructions are given for installing it on a linux system on the course website; if you aren't used to compiling/installing libraries on your own system, it might be quicker just to log in to daisy. Let us say your code file is named 'prog3.c' (if not, change the filename in the compile line). The code must compile and generate no warnings when compiled with:

```
gcc -Wall -std=c99 -g -L/home/phskak/Code/BLAS/ -L/home/phskak/Code/lapack-3.5.0/  \
    -I/home/phskak/Code/lapack-3.5.0/lapacke/include/ prog3.c -llapacke         \
    -llapack -lblas -lgfortran -lm
```

Be careful copying and pasting this command line: a script containing the compiler command is also on the website (for example if you add or remove spaces then this may prevent compilation). Please call the LAPACKE C interface and not LAPACK directly (because these are FORTRAN functions and the interface is not guaranteed to work the same way when compiled on different machines).

The lab computers require different libraries (the OS and hardware are different)

```
gcc -g band_utility.c -I/usr/include/lapacke -std=c99 \
     -Wall  -llapacke -llapack -lblas -lgfortran -lm
```

and this script is also on the moodle website next to the example. Note that you may also use the 'nenneke' unix machine to compile on as well as daisy: this has valgrind installed. You will need a CSC account: instructions are on the moodle page.

As before, you should test your code with simple testcases. I will, again, be marking it mostly based on how well it reproduces certain tests.

1 Frequency scan

It is usually much more interesting to know how the material will respond for a range of freqencies, rather than a single frequency.

The code therefore will scan over frequency ω, computing the displacement $h(x)$ for values ω_q, between ω_A (inclusive) and ω_B (exclusive), with integers $q \in [0, I-1]$. The value $\omega_q = \omega_A + q(\omega_B - \omega_A)/I$. Note that we expect the displacement to become large near resonant frequencies (if the damping is zero): the code should continue running if the matrix is singular, and it would be appropriate to output not-a-number values for the displacement in this case.

2 Suggestions on getting started

- There are analytical solutions to these equations for simple cases.
- The boundary conditions are really the main 'tricky' part of this problem. Think about how to simplify the problem to isolate this part of the problem.
- You can run a frequency 'scan' over only one value to keep things simple.

3 Specification

The input and output x grid has N grid points, which will exclude the right end of the domain; this grid will be taken to have spacing δx and run from $x_0 = 0$ to $x_{N-1} = L - \delta x$. The grid points you use in setting up the matrix problem are up to you. Ideally, for the sake of efficiency, the matrix should be a narrow banded matrix.

4 Breaking down the problem

You may find it helpful to first calculate the second derivative at each point and express equations based on that.

5 Input

The input parameters will be read from a file called 'input.txt'.

1. L: right x boundary of domain.
2. N: number of grid points
3. K: wavenumber of forcing.
4. γ: damping rate.
5. ω_A, minimum frequency in frequency scan.
6. ω_B, maximum frequency in frequency scan.
7. I, number of values in the scan over ω.

I will leave you to determine whether these should be read as integers or floating point numbers based on their meaning (note that the example input file does not necessarily have decimal points on all the values that should be floating point).

It is sufficient to use scanf to read in these parameters and simply test that the reads were successful: I am not specifically forbidding the use of other ways of reading input but I found roughly 50% of students attempting a more complicated input scheme got it wrong.

You may assume $N > 0$, $L > 0$, and $\mu, E > 0$.

The functions $\mu(x)$ and $E(x)$ (in that order) will be given at the N grid points as two columns of double precision numbers in a file called 'coefficients.txt'.

There are example input files and output file on the assignment page.

6 Output:

The code should output the frequency and spatial grid point, and the the real and imaginary parts of the displacement $h(x)$ to a file 'output.txt'. Each line of the file will contain the four values $\omega, x, \mathrm{Re}[h(x)], \mathrm{Im}[h(x)]$ for the grid points $x_0, x_1, ...x_{N-1}$, and for each value of ω in the scan (first, all the values associated with ω_0, then all for ω_1, etc.).

As usual, don't output any extra text to this output file.