

The Fast Fourier Transform (FFT) and Spectral Methods

HW 10: Wednesday, Nov. 17, 2021

DUE: Wednesday, Dec. 1, 2021

Thanksgiving is Wed. Nov. 24 to Fri. Nov 26

READ: Numerical Recipes in C++, Section 12.0-12.4 pages 600-627, on FFT's

OPTIONAL: FFTW documentation (pdf, etc. from www.fftw.org)

OPTIONAL: Landau, Paez, and Bordeianu: section 10.8,9 on FFT, chp. 14 on parallel programming

OPTIONAL: Newman, Discrete Fourier Transforms, chp. 7

1. **FINAL PROJECT OUTLINE DUE ON TUE. NOV. 23, 2021:** Please write a one page outline of your final project. This should include a title and a few paragraphs and equations outlining your project goals and how you expect to accomplish them. You should also include at least one reference (more than just wikipedia). Your outline must be approved before turning in your final project report. You may start to work on the project before your outline is approved but you may need to change your project if the outline is not accepted. Also, if this is a joint project (with another course) you should state this in your outline and on your final report.

2. **FINAL PROJECT DUE ON WED. DEC 15, 2021** (during exams): Please read the previous handout. This will be graded out of 20 points total. Because of the tight deadlines for senior grades etc. there will be a 2 point/day penalty for late reports. If you have another exam on this day your project is due the next day you do not have an exam (please see me in advance). Please plan to work on your project well in advance.

3. Please submit your final project on Canvas and include the source code in a separate file that I can compile if necessary. Have a nice winter vacation.

4. **PROJECT PRESENTATIONS** Graduate students enrolled in AEP5380 should plan to give a short (about 10 min) oral presentation on their project proposal during the last week or so of class. If possible to the class, but may be individual or small group presentations if necessary. Schedule to be arranged.

5. For your information the homework mean and standard deviation are 8.7 and 0.9 (excluding late HW, not yet graded) for homework number 1 through 7.

PROBLEM (10 points):

A. The FFTW Subroutine Package

The FFT (Fast Fourier Transform) algorithm is one of the fastest numerical methods available. Many problems are rephrased to use Fourier transforms to take advantage of this method. The popular FFTW package (www.fftw.org, the "Fastest Fourier Transform in the West"[3]) is one of the fastest and most sophisticated subroutine packages available. It is also free (under the GNU public license, GPL) and fairly well documented. It is used in most software packages. There are pre-compiled subroutine libraries (in binary) for most computer platforms so you don't need to compile it (its long). Please refer to the FFTW documentation.

The speed of execution of many FFT's (particularly multidimensional FFTs) is as much dependent on the memory bandwidth as the number of floating point operations. FFTW uses a specialized memory allocation scheme to align the memory properly. It has its own plain C-like memory allocation routine `fftw_malloc()` and data types which should be used. FFTW has a unique approach to obtaining maximum speed for repeated execution of the same size FFT (common in many large simulation). FFTW first tries (at run time) many different forms of FFT's to find the style that is the fastest on the computer its running on and for the size being used (saved as a plan). This step may take significant time (only done once for each size and type), but if the same size FFT is performed many times, FFTW can yield an overall improvement of a

factor of 2X or 3X in many calculations. This HW problem is too simple to see this improvement, but larger calculations can see a significant improvement. You can also use the FFTW_ESTIMATE mode in place of the FFTW_MEASURE mode to avoid this planning step and just guess at a good FFT style (faster overall if only one FFT is performed).

B. FFT in python

Sadly, scipy does not yet use FFTW, but has another FFT that is fast but not quite as fast. The module pyFFTW (not included in anaconda) can be installed separately to use FFTW. An example of some of the typical steps in using the scipy 2D fft are:

```
import numpy as np
from scipy.fft import fft2, ifft2
:
# psi and psi2 are 2D complex np arrays
:
psi2 = fft2( psi ) # forward transform
:
psi = ifft2( psi2 ) # inverse transform
:
```

C. Spectral Methods

Some partial differential equations with simple boundary condition (and constant coefficients) can sometimes be solved using what are called *spectral methods* (see for example section 6.4 of [4]). This means that a solution is developed in Fourier Transform space and then transformed back into real space. The Fast Fourier Transform (or FFT) is an efficient numerical method to calculate Fourier Transforms and will be used in this homework assignment.

Consider the 2D wave equation for a displacement $\psi(x, y)$ as a function of position $\vec{x} = (x, y)$ and time t :

$$v^2 \left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right] \psi = \frac{\partial^2 \psi}{\partial t^2} \quad (1)$$

where v is the velocity of the wave propagation. This equation might model the behavior of a water wave, elastic membrane or electromagnetic wave. Express the solution as a Fourier series in the form:

$$\psi(\vec{x}, t) = \sum_{ij} a_{ij}(t) \exp(-i\vec{k}_{ij} \cdot \vec{x}) \propto FT_{\vec{x}}^{-1}(a_{ij}) \quad (2)$$

where $\vec{k}_{ij} = (k_{xi}, k_{yj})$ is a 2D spatial frequency vector ($\sum_{ij} = \sum_i \sum_j$). This uses the traditional physics placement of the 2π factor but remember that the FFT uses an explicit 2π as $\exp(2\pi i \vec{k}_{ij} \cdot \vec{x})$, and the frequencies comes out in a rearranged order. Insert this expression into the original equation (1):

$$-v^2 \sum_{ij} k_{ij}^2 a_{ij}(t) \exp(-i\vec{k}_{ij} \cdot \vec{x}) = \sum_{ij} a_{ij}''(t) \exp(-i\vec{k}_{ij} \cdot \vec{x}) \quad (3)$$

Equating coefficients of $\exp(-i\vec{k}_{ij} \cdot \vec{x})$ yields the solution:

$$-v^2 k_{ij}^2 a_{ij}(t) = a_{ij}''(t) \quad (4)$$

$$a_{ij}(t) = b_{ij} \exp(+iv|\vec{k}_{ij}|t) + c_{ij} \exp(-iv|\vec{k}_{ij}|t) \quad (5)$$

$$\psi(\vec{x}, t) = \sum_{ij} \left[b_{ij} \exp(+iv|\vec{k}_{ij}|t) + c_{ij} \exp(-iv|\vec{k}_{ij}|t) \right] \exp(i\vec{k}_{ij} \cdot \vec{x}) \quad (6)$$

where b_{ij} and c_{ij} are complex constants independent of time. If the displacement is released from a specified position initially at rest:

$$\frac{\partial \psi(\vec{x}, t=0)}{\partial t} = 0 = \sum_{ij} [b_{ij} - c_{ij}] (iv|\vec{k}_{ij}|) \exp(i\vec{k}_{ij} \cdot \vec{x}) \quad (7)$$

This restricts the solution to $b_{ij} = c_{ij}$ and:

$$\psi(\vec{x}, t) = \sum_{ij} d_{ij} \cos(v|\vec{k}_{ij}|t) \exp(i\vec{k}_{ij} \cdot \vec{x}) \quad (8)$$

$$= FT^{-1} \left[\Psi(\vec{k}_{ij}, t=0) \cos(v|\vec{k}_{ij}|t) \right] \quad (9)$$

$$= FT^{-1} \left[FT[\psi(\vec{x}_{ij}, t=0)] \cos(v|\vec{k}_{ij}|t) \right] \quad (10)$$

where $d_{ij} = 2b_{ij}$. To solve this equation using a spectral method, first calculate the $d_{ij} = FT[\psi(\vec{x}, t=0)]$ coefficients from a given initial shape $\psi(\vec{x}, t=0)$ using a discrete fast Fourier transform (FFT). Then multiple these d_{ij} values by the above factor and calculate $\psi(\vec{x}, t > 0)$ at some later time using an inverse FFT.

In this homework problem assume $L_x = L_y = 1000$ m., the speed of sound $v = 343$ m./sec, $N_x = N_y = 512$, and set the initial displacement to be two small Gaussians near the center (to watch interference):

$$\psi(\vec{x}, t=0) = \sum_{n=1}^2 A_n \exp \left[-\frac{|\vec{x} - \vec{x}_n|^2}{s_n^2} \right] \quad (11)$$

where $x_n = (0.4, 0.5)L_x$ and $y_n = (0.4, 0.6)L_y$ in m, and $s_n = (10, 20)$ in m. The amplitudes should be $A_n = (1, 2)$.

For simplicity treat ψ as complex although its is initially real (FFT indexing is easier). Display the real part of $\psi(\vec{x}, t)$ for times 0.0, 0.2, 0.4 and 0.8 sec. You may display the results as a contour plot (as in HW 6) or as a gray scale image which is easier to interpret.

In python:

```
import numpy as np
import matplotlib.pyplot as plt
#
img = plt.imshow( np.real(psi), aspect='equal', extent=(0,Lx,0,Ly) )
#img.set_cmap( 'gray' )
img.set_cmap('hot') # works OK
plt.colorbar()
plt.title('result')
plt.savefig('result_py.eps')
plt.show()
```

HINT: If your results look "scrambled", try taking the FFT of just $\sin(kx)$ and verify the frequencies are where you think they are. Remember that the frequencies come out in "rearranged" order (0, 1, 2, 3, ... (N/2-1), -N/2, ... -3, -2, -1) with $\Delta k_x = 1/L_x$. It is sometimes easier to pre-calculate an array of k_x values and an array of k_y values at the beginning for later use.

OPTIONAL-1: Try using the fast sin transform which will impose a constraint on the end points and use only about half of the floating point operations.

NOTE: This can be animated to make an interesting presentation.

References

- [1] E. O. Brigham, *The Fast Fourier Transform and Its Applications*, Prentice Hall, 1988.
- [2] J. W. Cooley and J. W. Tukey, Math. Computation 19 (1965) p. 297-301

- [3] Matteo Frigo and Steven G. Johnson, "Design and Implementation of FFTW3", Proc. of the IEEE, 93 (2005), p. 216-231.
- [4] N. J. Giodano and H. Nakanishi, *Computational Physics, 2nd edit.*, (Prentice-Hall 1997,2006).
- [5] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery *Numerical Recipes, The Art of Scientific Computing, 3rd edit.*, Camb. Univ. Press 2007.
- [6] James S. Walker, *Fast Fourier Transforms, 2nd edit.*, CRC Press, 1996.