PY 421 - Introduction to Computational Physics

Homework # 4. February 21, 2013.

Due at discussion time (3PM) on Friday March 1.

The problem.

Assignment #4 consists in writing a code which uses the fast Fourier transform to "unblur" a blurred picture. The program should asks the user to input the number n of preliminary blurring steps he/she wants to do and the blurring parameter. After the image has been blurred, the program should ask the user to input the number n' of unblurring steps he/she wants to perform and the program should then use the FFT to perform the requested unblurring all at one time. If the user inputs a number $n' = n$ the unblurring should reproduce the original image.
For your convenience I have posted on the courseware/code directory a program called "image_unblur.f90" which contains all the code needed for the assignment apart from the actual unblurring operations. The code also declares variables which you may find convenient to use, namely:

- `fc`: a complex valued two dimensional array with the same dimensions as `f`. `f` should be copied to `fc` because the subroutine which performs the FFT expects a complex argument. The instruction `fc=f` has been left in the code.
- `job`: an integer number passed to the subroutine "fftr" which should take value 1 for the direct FFT, -1 for the inverse FFT.
- `K`: An integer parameter equal to 8 which should be passed to "fftr". The image size is $2^K \times 2^K$.
- `SIZE`: An integer parameter equal to $2^K = 256$.
- `x y`: Two integer variables which may be used as loop indices.
- `cosk`: An array of dimension (0:SIZE-1) where I store $\cos(2\pi j/SIZE)$ for $j = 0, N-1$, for convenience.
- `lambda`: An array of dimension (0:SIZE-1,0:SIZE-1) where I store the eigenvalues of the blurring operator.

- `nstp2`: The desired number of unblurring steps.
- `EPS`: A parameter which I set equal to 10^{-10} and I use to cut-off from the unblurring the modes which have fallen below this value (see below.)

Please note that you do not have to use the same names as above for the variables in your code. All those variables have been declared in my own solution to the problem and I left them in the program for your convenience. The program "image_unblur.f90" also contains the subroutine "fftr" which performs the FFT.
Before starting your work, you should copy from the courseware/code directory the latest version of the file "makefile.imblur" and check that the command

```
make -f makefile.imblur image_unblur
```

produces the executable "image_unblur" Of course "image_unblur" will not unblur the image, but you can start building your code from here and use make often to make sure that the code you have written up to then still compiles.
The theory behind the algorithm is as follows. We know that if we expand the image into eigenvectors of the discretized Laplacian the expansion coefficients c_{k_x,k_y} undergo a change $c_{k_x,k_y} \to (1-\lambda_{k_x,k_y}\,dt)c_{k_x,k_y}$ at each blurring steps and therefore that their final value $c^{(n)}_{k_x,k_y}$ after n blurring steps is related to their initial value $c^{(0)}_{k_x,k_y}$ by

$$c^{(n)}_{k_x,k_y} = (1-\lambda_{k_x,k_y}\,dt)^n c^{(0)}_{k_x,k_y} \tag{1}$$

Thus your code should use an FFT to calculate the coefficients $c^{(n)}_{k_x,k_y}$ in the expansion of the final image, after the blurring, and implement the opposite transformation

$$c^{(n)}_{k_x,k_y} \to c'_{k_x,k_y} = c^{(n)}_{k_x,k_y}/(1-\lambda_{k_x,k_y}\,dt)^{n'} \tag{2}$$

You can take advantage of the code in fftr_use_2d.f90 and in fftr.f90 to calculate the coefficients $c^{(n)}_{k_x,k_y}$ from the blurred image and to reconstruct the final image from the coefficients c'_{k_x,k_y}.
A note of caution: as the image blurs, the coefficients c_{k_x,k_y} decrease in magnitude. If the blurring is so pronounced that most of the coefficients have fallen in value below the magnitude of double precision numerical errors,

then the image can no longer be unblurred. If some but not most of the coefficients have fallen in value below the magnitude of the numerical errors, trying to bring these coefficients to their original value by dividing them by $(1 - \lambda_{k_x,k_y}\, dt)^n$ will produce amplified numerical errors, rather then the original coefficients, and the reconstruction of the image will produce a wrong result. Thus within the range of blurring where most, but not all of the Fourier components of the image can be reconstituted a good strategy consists in applying the transformation of Eq. 2 only to those coefficients whose magnitude is somehow larger than typical round-off errors, for example larger than 10^{-10}.
You should call your program image_unblur.f90 and you should copy it to the file

~rebbi/courseware/asgn/asgn4.xxyyyy

where xxyyyy stands for your personal code.

Grading criteria.

100 points will be awarded for a well written, correct code which unblurs the picture as expected. Points will be taken off for errors or poorly written code, accordingly to the severity of the code's shortcomings.