PY 421 - Introduction to Computational Physics

Homework # 6. March 21, 2013.

Due on Friday March 29 at discussion time (3PM).

For this assignment you will have to complete and return three programs and execute them collecting data which will also have to be returned. The assignment is divided into problems, which explain the various tasks you will have to do and will be scored separately.

Problem 1

You will find in the ∼rebbi/courseware/code/ directory the following files, which you should copy to your working directory:
relax_driver.c
relax_gj.f90
relax_gs.f90
relax_ev.f90
makefile.relax
All these programs are related to the problem of calculating an electrostatic potential in presence of conductors illustrated in the file "project2.pdf".
The program "relax_driver.c" is similar to the program "image_blur_driver.c" and its purpose is to display the potential field during the iterations of the relaxation algorithms. "relax_driver.c" receives the data which must be displayed from the subroutine relax contained in the f90 files.
The file "makefile.relax" contains the rules for compiling all the programs in this assignment.
For this first problem you must complete the programs "relax_gj.f90" and "relax_gs.f90" so that the executables implement the Gaus-Jacobi and Gauss-Seidel relaxation algorithms for the calculation of the potential. The geometry of the problem is the one illustrated in the lecture notes "project2.pdf". Most of the code is already contained in the files you are given. In particular, the programs contain a subroutine "init" which is called during the initialization phase of the calculation which returns a logical mask "inside" which takes value True outside of the conductors, i.e. in the region where the potential must be calculated, and False inside the conductors. "init" also returns an initial potential "v0" which takes some definite values on the

conductors (printed out at the beginning of the program) and value 0 in the region where the potential must be calculated. Your task is limited to writing the instructions which implement the actual iterations of the algorithm, the instructions that calculate the residue, and the instructions to print out the current total number of steps and the residue and to write the current total number of steps and the logarithm of the residue on the file "relax.data".
Recapitulating the algorithms of Gauss-Jacobi and Gauss-Seidel, in both cases at each relaxation step the current value $v(x, y)$ of the potential should be replaced by the average of the neighbors:

$$v(x,y) \to v(x,y)' = [v(x+1,y)+v(x-1,y)+v(x,y+1)+v(x,y-1)]/4 \quad (1)$$

The replacement should of course occur only for the values of x, y which label points inside the domain delimited by boundaries and outside the circle and the rectangle. The difference between the two algorithms is that in the algorithm of Gauss-Jacobi the replacement must be done globally, while in the algorithm of Gauss-Seidel it is done point by point. For the program implementing the Gauss-Seidel relaxation procedure, which depends on the ordering, you should upgrade the variables serially in a double loop, with the variables x (horizontal coordinate) incremented in the inner loop, the variable y in the outer loop. All floating point variables should be in double precision.
The residue is defined by

$$r = \sqrt{\frac{\sum_{x,y\in \text{inside}} w(x,y)^2}{N_{\text{inside}}}} \quad (2)$$

where

$$w(x,y) = v(x,y) - [v(x+1,y)+v(x-1,y)+v(x,y+1)+v(x,y-1)]/4 \quad (3)$$

$x, y \in$ inside stands for the points inside the domain, and N_{inside} is the total number of points inside the domain.
You may compile the programs that implement the two relaxation algorithms by using the makefile "makefile.relax" with the commands
make -f makefile.relax relax_gj
or
make -f makefile.relax relax_gs
Note that the programs "relax_gj.f90" and "relax_gs.f90" which you will find in the ∼rebbi/courseware/code/ directory will compile and execute without

errors, but of course the executables will not implement the algorithms and will only print and write to file the initial information.

For this part of the assignment you must return the programs relax_gj.f90 and relax_gs.

Problem 2

Run the programs relax_gj and relax_gs with sizes 128 and 256, with number of images equal to 500, 40 relaxation steps per image and 5 images per print. The potential should be set equal to 20 on the circle, -20 on the rectangle, -10 on the upper boundary, 10 on the lower boundary and 0 on the left and right boundaries, as already done by the subroutine init. Plot the curves giving the logarithm of the residue (as defined in Eq. 2) versus total number of steps in a single postscript file, called relax.ps, and return this file. Make sure that the four graphs are labeled in an understandable manner (for instance, you can save the data in files called relax.gj128, relax.gs256 etc., and the names of the files will then serve to label the plots). If you prefer, you can plot all four sets of data in a single figure, making sure however that the graphs corresponding to the different data points are clearly labeled.

Problem 3

Complete the program "relax_ev.f90" (ev for eigenvector), so that it calculates the lowest and highest eigenvalues, $\lambda_{\min}$ and $\lambda_{\max}$ of the operator $A = -a^2\Delta_L/4$, as well as the corresponding eigenvectors, proceeding as described below. Note: the lattice spacing a never enters in the code. It is used in the equation above because the expression for the discretized Laplacian contains a^2 at denominator, which is then canceled in the expression for A. The action of A on a vector of components $v(x, y)$ is given by:

$$(A\,v)(x,y) = v(x,y) - [v(x+1,y) + v(x-1,y) + v(x,y+1) + v(x,y-1)]/4 \quad (4)$$

A acts only inside the region delimited by the conductors, i.e. on the components of v with indices x, y for which the logical variable $inside(x, y)$ is true, and satisfies Dirichlet boundary conditions, i.e. the values of v falling outside the region of definition should be taken equal to 0.
In order to find the minimum and maximum eigenvalues and the corresponding eigenvectors consider the iterations

$$v \to v' = v - \omega A\,v \quad (5)$$

starting from an arbitrary initial value for v, and where ω is a suitable parameter. The spectrum of A, i.e. the set of its eigenvalues λ, is contained between 0 and 2. Thus, if we take $\omega < 1$ all the eigenvector components of v will decrease, but at different rates and eventually the eigenvector component corresponding to the lowest λ will dominate. If in the course of the iterations we normalize periodically the vector v, for example in such a way that the absolute values of its component range up to 1 included, after a sufficient number of iterations v will give a very good approximation to the eigenvector with the lowest λ.
If, instead, we take ω sufficiently larger than 1, then some of the eigenvector components will increase in magnitude during the iterations, with the eigenvector component with largest λ growing faster than all others. Thus, if we periodically renormalize the vector v as described above, after a sufficient number of iterations v will give a very good approximation to the eigenvector with the lowest λ.
Thus the completed program "relax_ev" will work as follows. The user is asked for the size of the domain, the total number of images to be displayed, the number of relaxation steps between images and the value of ω. The program will call the "init" subroutine to get the logical array "inside". It then sets the initial value of v equal to 0 outside the domain of definition of A and 1 inside. At this point the program implements the relaxation iterations until the desired number of images is reached.
After the relaxation steps have been performed and before returning control to the driver, "relax_ev" must normalize v. This can be conveniently done by using the statement `MAXVAL`, for example with

```
vmax = MAXVAL(ABS(v),MASK=inside)
```

(the mask is really not necessary, since v will be 0 elsewhere)

```
v=v/vmax
```

Then the program estimates the eigenvalue by calculating

$$\lambda = \frac{vAv}{v^2} \tag{6}$$

This can also be implemented very easily with Fortran statements. For example one can calculate $w = A\,v$ and then

```
lambda= SUM(v*w,MASK=inside)/SUM(v*v)
```

The program should print now the total number of steps done so far and the estimate of λ.

Finally since a vector ranging between -1 and 1 would not produce much color variation in the display, the program resets the value of the potential by multiplying it by -20 before returning its value to the driver.
As a last step, when the desired number of images has been obtained, the code prints out one line of the array `v` as follows:

```
PRINT '(8F9.4)',v(:,32)
```

The completed program relax_ev.f90 should be returned together with relax_gj.f90 and relax_gs.f90

Problem 4

For this problem you should run first the `relax_ev` program with size 128, total number of images 200, and number of steps between images 250. You should run the program with $\omega = 0.95$ to estimate the lowest eigenvalue $\lambda_{\min}$ and $\omega = 1.05$ to estimate the highest eigenvalue $\lambda_{\max}$. You should write the values you thus found for $\lambda_{\min}$ and $\lambda_{\max}$ in the file relax.ps
Plot again, in a separate graph, the data from the Gauss-Jacobi iterations with size 128. The theory of the algorithm shows that the error and the residue should decay asymptotically with the number n of steps as

$$r(n) \approx (1 - \lambda_{\min})^n = e^{n \log(1-\lambda_{\min})} \approx e^{-n\lambda_{\min}} \tag{7}$$

Plot on the same graph a line

$$r = c - n\lambda_{\min} \tag{8}$$

using for $\lambda_{\min}$ the value you found in problem 3 and adjusting the constant c by trial and error so that the line fits the asymptotic behavior of the graph of the residue. This plot should also be included in the file relax.ps

Completing and returning the assignment.

For this assignment you must return the files relax.ps (see problems 2 and 4) and the programs relax_gj.f90, relax_gs.f90 and relax_ev.f90
These files must be put together in a single file with the tar command (do not compress the resulting file) and the tar file should be copied on the CAS cluster to the file

~rebbi/courseware/asgn/asgn6.xxyyyy,

where xxyyyy stands for your identifier, which will be given to you separately. You can overwrite the file asgn6.xxyyyy as many time as you like until the deadline for the assignment. After the deadline, the write permission on the file will be revoked.
Please note that, although when you read the text of this assignment it may look rather long, the actual work you should do for it is not a lot. Completion of the three programs requires writing only a few lines of code and you are not asked to produce a full fledged report, but only to incorporate a few graphs in a single document. The assignment is instructive though and, if you have the time, I encourage you to run numerical experiments beyond what is strictly required for this assignment.

Bonus Problem - worth 10 points to be applied to one score of your choice, including the first midterm, up to this assignment. Collaborative work is not allowed for this problem.

From problem 4 you may have noticed that $\lambda_{\min} + \lambda_{\max} = 2$ and that the corresponding eigenvectors are related by $v(x,y)_{\max} = (-1)^{x+y} v(x,y)_{\max}$. Give a clear demonstration that the above relations are general, namely that the eigenvectors of A satisfy the following property:
For every eigenvector $v(x,y)$ there exits an eigenvector $v'(x,y)$ related to the former by $v'(x,y) = (-1)^{x+y} v(x,y)$. Moreover, the corresponding eigenvalues are related by $\lambda + \lambda' = 2$.

Grading criteria.

25 points will be awarded for a correct solution to each problem, for a maximum score of 100 points. Solutions with errors or poorly written code will be given partial credit, according to the severity of the error.