UNIVERSITY OF Southampton

SESG6025 Coursework 2013/14: PDE methods

Due: Wednesday 8th January 2014 (12 midday). 50% weighting.

Aims: In this coursework you will investigate methods for solving PDEs. The preferred language for submission is Python. Solution may be done in other languages (e.g. C)

Objectives: To develop solvers using the course notes & references and further investigate them when solving a partial differential equation.

Requirements and hand-in: You should email **two** files (together as attachments to a single email) to sesg6025@soton.ac.uk with the email subject "**Coursework**"

Files to email in: (1) The filename for the code (not an executable) must be "**multi.py**"

(2) The write-up in a single pdf "**multi.pdf**"

The single "**.py**" file should set up and solve the PDE as per each question and when executed and leave the answer to questions (1 - 4) as a single line of output each to the screen appropriately labelled. You should clearly label the functions which you write in each question in the code. The "**.pdf**" file should give the answer to question (5).

*[Special note for submission in other languages please add a 3rd file to your submission called e.g. "**multi.c**", and ALSO submit an empty "multi.py" file, and let me know by email.]*

Questions

Consider solving the PDE (which we refer to as PDE (1) throughout the questions)

$$\nabla^2 u = \rho, \qquad \text{PDE (1)}$$

where $u = u(x, y)$, and $\rho(0.5, 0.5) = 2$ where $0 \le x, y, \le 1$, and $u = 0$ on the boundaries.

In each of the questions (1-4) marks will be awarded for the code, implementation and the write-up in question (5) demonstrating systematic testing and explanation. The course notes and references [1, 2] may be helpful. For each of questions (1-4) the code, when executed, should produce a single output to the screen which checks whether the solution obeys $\nabla^2 u = 2$ at (0.5, 0.5).

1) Develop an enhanced and debugged implementation of the code in section 4 of the notes to solve the above partial differential equation PDE (1). The code, when executed, should produce a single output which checks whether the solution obeys $\nabla^2 u = 2$ at (0.5, 0.5).

(20 marks)

2) Instead of using a built-in solver method, implement and use a successive over-relaxation solver within your code to solve PDE (1) – see section 2 of the notes for the formula, and include in your write-up consideration of the parameter ω. Remember to give a single output for the check.

(20 marks)

3) In the notes we considered the *first central difference approximation* for the *second* derivative of $f(x)$. It is possible to derive higher order derivatives to various orders of accuracy. Replace the simple 4 point stencil with the following stencil in your code for question (1) and solve PDE (1). Remember to give a single output for the check.

$$f''(x) = \frac{-f(x-2h) + 16f(x-h) - 30f(x) + 16f(x+h) - f(x+2h)}{12h^2} + O(h^4) \qquad (2)$$

(10 marks)

4) In computational methods, it is often necessary to investigate and implement alternative methods either to use, or as an independent check for a code that you are working on. For this question, you need to investigate and find-out about an enhancement to the Gauss-Seidel method often called the "red-black" formulation. It is also sometimes used as a demonstration for parallel programming in high performance computing. Develop a Gauss-Seidel "red-black" solver and use it to solve PDE (1). You may find refs [1,2] helpful. Remember to give a single output for the check.

(15 marks)

5) Give a short write-up for the codes you have written in questions 1-4 (**no more than 6 sides**) demonstrating systematic testing and explaining and justifying what you have done, showing e.g. convergence of the methods and other relevant information. You should also include correctly and clearly labelled axis plots for the results from the questions. Where appropriate, compare the methods. Comment on how you might speed-up your code further. Include in your write-up a brief comparison with the multigrid method explained in reference [2]: you do NOT need to implement code for this.

(35 marks)

Prof Simon Cox, sjc@soton.ac.uk. Building 25/2037 Phone Ext 23116.

References

You may find these references useful:

[1] From http://www-users.cs.umn.edu/~saad/books (last checked Oct 13)

Saad, Y. "Iterative Methods for Sparse Linear Systems" (2003)

http://www-users.cs.umn.edu/~saad/IterMethBook_2ndEd.pdf

[2] Press, WH, Teukolsky, SA, Vetterling, WT, and Flannery BP (1992, 1996, 2007, and later) "Numerical Recipes in C" , "Numerical Recipes in Fortran", "Numerical Recipes 3rd Edition". These books contain both code and algorithms.

See www.nr.com for more details: you can also read these books (2nd edition) online for free (you may need to install an Adobe plugin). Note that the 2nd edition code in these references is old and uses "1 based" indexing for the arrays- this was updated in the 3rd edition code.

http://www.nrbook.com/a/bookcpdf.php

http://www.nrbook.com/a/bookcpdf/c19-0.pdf

http://www.nrbook.com/a/bookcpdf/c19-5.pdf

http://www.nrbook.com/a/bookcpdf/c19-6.pdf

Update to notes: Please use this in place of Figure 7 in the notes (interchange of *i* and *j*)