UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR11174 NUMERICAL ALGORITHMS FOR HIGH PERFORMANCE COMPUTING

Tuesday 11 th December 2018

14:30 to 16:30

INSTRUCTIONS TO CANDIDATES

1. **Note that ALL QUESTIONS ARE COMPULSORY.**
2. **EACH QUESTION IS WORTH 25 MARKS. Different sub-questions may have different numbers of total marks. Take note of this in allocating time to questions.**
3. **CALCULATORS MAY BE USED IN THIS EXAMINATION.**

EPCC Courses

Convener: M.Mistry
External Examiner: Matt Probert

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. A system of N simultaneous equations can be expressed as a dense linear system

$$Ax = b$$

where A is an $N \times N$ matrix of known coefficients, b is a vector of length N of known values, and x is a vector of length N of unknown values to be determined. Such a system can be solved via LU factorisation where the factorisation is usually done *in place*.

(a) Explain what is meant by "in place" in this context and why you might want to do the factorisation in place. [*3 marks*]

(b) When carrying out the factorisation, *partial pivoting* is usually used. Explain what partial pivoting is, why it is used, and why it is important to store the information about the pivoting carried out on the matrix. [*6 marks*]

(c) Explain the main computational steps involved in using LU factorisation to solve the system and state the cost associated with each step as a function of N using "big O" notation. [*7 marks*]

When attempting to model a physical system, a mathematical model of the system is usually created which leads to a numerical algorithm which may finally be implemented as a piece of code. The code may then be run with a given input leading to a set of results. These results will not in general be accurate.

Explain the three main sources of error, and how they arise and what can be done to mitigate these. [*9 marks*]

2. (a) i. State the four key stages of the general N-body simulation algorithm in order, describing when they occur during the simulation. [*6 marks*]

ii. For the second and third stages of the N-body algorithm where the particles move, explain how the number of operations $T(N)$ scales with the number of particles N and state this using "big O" notation. You may find it useful to consider what loop (or loops) you would need to write for each stage. [*6 marks*]

iii. Based on your answers to part ii, which stage or stages dominate the overall number of operations required by the N-body algorithm? How does the overall runtime of the N-body algorithm therefore scale with N? [*3 marks*]

(b) N-body simulations are based on Newton's equations of motion, which require a discretisation scheme to be solved numerically. The simplest scheme is the Euler scheme, which says that new positions and velocities of particles after a time step δt can be computed using:

$$x(t + \delta t) = x(t) + v(t) * \delta t \quad (1)$$
$$v(t + \delta t) = v(t) + a(t) * \delta t \quad (2)$$

From the derivation of the Euler scheme we know that it is more exact to say:

$$x(t + \delta t) = x(t) + v(t) * \delta t + O(\delta t^2) \quad (3)$$
$$v(t + \delta t) = v(t) + a(t) * \delta t + O(\delta t^2) \quad (4)$$

i. Explain what is meant by the *local* error. By comparing the equations (3), (4) to the Euler scheme equations (1), (2), how do the local errors in the Euler scheme for the computed positions and velocities depend on the time step δt? [*5 marks*]

ii. Explain what is meant by the *global* error of a discretisation scheme. By considering n time steps, each of duration δt, covering a total simulated time T, work out the dependence on δt of the global errors in the positions and velocities at time T for the Euler scheme. [*5 marks*]

3. The Discrete Fourier Transform (DFT), F_n, of a set of N complex points f_k is defined as

$$F_n = \sum_{k=0}^{N-1} f_k e^{2\pi i k n/N}, \text{ for } 0 \leq n < N.$$

(a) If this sum is worked out explicitly for each of the N points, what is the computational cost in terms of N? Explain your answer. [*2 marks*]

(b) If instead the calculation is performed via the FFT algorithm, explain what the computational cost is in terms of N, justifying this by giving a brief overview of the algorithm. [*6 marks*]

(c) Imagine you have a 3-dimensional computational fluid dynamics problem of size 256^3 and you are required to carry out the 3D FFT of the entire array. How many 1D FFTs are required to compute this? [*4 marks*]

(d) Imagine you are going to perform the above 3D FFT in parallel on a distributed memory machine, where the data is decomposed across processors using a 1D processor grid. With this decomposition each processor holds several 2D planes of data. Describe, using diagrams where appropriate, the operations involved in carrying out this 3D FFT, indicating clearly where local FFTs and data transpose operations (communications between processors) take place. You should assume you have access to an efficient serial FFT implementation, e.g. the FFTW library. [*7 marks*]

(e) How is the scalability in terms of the number of processors limited with a 1D processor decomposition and what could be done to improve this? Assume here that the problem size is as above (i.e. 256^3). [*6 marks*]

4. Krylov subspace methods such as Conjugate Gradient (CG) and Bi-conjugate gradient with stablisation (BiCGstab) are known as *iterative methods* and are often used to solve large sparse linear systems of the form

$$Ax = b.$$

A 1D pollution problem is an example of such a system and can be modelled using the following discrete equation

$$-u_{i-1} + (2 + ah)u_i - (1 + ah)u_{i+1} = 0$$

where u_i are regularly-spaced discrete values of the pollution density, a is the wind strength, $h = 1/(M + 1)$ and M is the number of discrete points.

(a) Given the matrix-vector multiplication operations carried out within each CG or BiCGstab iteration are expensive, this operation is often carried out in parallel. Assume here that the rows of the matrix A are distributed across processors and the vectors are replicated across all processors. For simplicity assume here that no sparse storage scheme is used. Explain using pseudocode and/or diagrams where necessary the steps involved to perform the matrix-vector muliplication in parallel at each iteration. [*9 marks*]

(b) How would this code differ if the rows of each vector were distributed across processors instead? [*6 marks*]

(c) If the distributed matrix were instead stored using a sparse storage scheme rather than as a dense matrix, why might it be important in terms of memory that vectors are distributed across processors if the problem size is large? [*3 marks*]

(d) Given the choice between using CG or BiCGstab, which would you choose? Include in your answer the cases where there is no wind (i.e. $a = 0$) and when a is either small (i.e. light wind) or large (i.e. strong wind). [*7 marks*]