This question paper consists of 7 printed pages, each of which is identified by the Code Number COMP392001.

School of Computing

January 2017

COMP392001

Parallel Scientific Computing

Answer ALL THREE questions

Time allowed: 2 hours

TURN OVER FOR QUESTIONS

Question 1

(a) (i) What is the difference between weak and strong scaling? **[1 mark]**

(ii) In MPI, what is the difference between blocking and non-blocking communication? **[1 mark]**

(b) Fig. 1 gives an MPI-C code segment for the forward elimination phase of Gaussian elimination, in which row `i` is subtracted from each lower row `j>i` to eliminate the values in column `i`. Inspect the code and then answer the following questions.

(i) This example partitions the matrix rows into strips. What is the alternative form of row partitioning known as, and why might it give better performance for this calculation? **[2 marks]**

(ii) What is the purpose of the `MPI_Send` and `MPI_Recv` in this code segment to the correct functioning of the algorithm? **[1 mark]**

(iii) Someone suggests replacing the blocking `MPI_Send` with a non-blocking `MPI_Isend`. Why might this improve performance? Where should the corresponding call to `MPI_Wait` be placed? **[2 marks]**

(iv) In the serial version of the backward substitution phase, the transformed matrix is solved row by row starting from the bottom, finding one new unknown for each row. Why might this present a challenge to parallelise? Briefly outline how it might be solved in MPI using the method known as pipelining. **[3 marks]**

(c) You are given serial code that takes two matrices A and B, both with n rows and m columns, and adds them to give a third matrix C of the same size. You are asked to convert this to use p parallel processes in MPI.

(i) What is the time $t_{\rm s}$ for the serial code to execute, on a time scale in which one floating pointer operation takes one time unit? You do not need to consider memory allocation, memory access, or any integer arithmetic. **[1 mark]**

(ii) For the MPI version, the matrices A and B are initially defined on rank 0 only. Suppose each matrix is partitioned between the p processes using a single point-to-point communication for each matrix row. Assuming n is a multiple of p, show that this communication time $t_{\rm comm1}$ is

$$t_{\rm comm1} = 2\frac{n}{p}(p-1)(t_{\rm startup} + mt_{\rm data}) \; .$$

Explain your reasoning. **[4 marks]**

(iii) To complete the algorithm, each of the rows of matrix C stored on the individual processes are sent to rank 0, again using a single point-to-point communication per row. Determine the parallel execution time $t_{\rm p}$ and hence the parallel efficiency $E = t_{\rm s}/(pt_{\rm p})$. **[3 marks]**

(iv) Consider the case of a square matrix $n = m$. For n and p both large, does any combination of n and p give rise to isoefficiency? What does this tell you about parallelising matrix addition in this manner? **[2 marks]**

[question 1 total: 20 marks]

CONTINUED ON FACING PAGE

```
...  /* Distribute matrix A to all ranks.  Vector b remains on rank 0.*/
for( i=0; i<n-1; i++ )
{
     ...  /* Broadcast row_i to all ranks */
     for( j=i+1; j<n; j++ )
     {
          if( rank==j/rowsPerProc )
          {
              local_i = j % rowsPerProc;
              mult = localRows[local_i][i] / row_i[i];
              if( rank!=0 )
              {
                  MPI_Send(&mult,1,MPI_DOUBLE,0,...);
              }
              for( k=i; k<n; k++ )
              {
                  localRows[local_i][k] -= mult*row_i[k];
              }
          }
          if( rank==0 )
          {
              if( j/rowsPerProc>0 )
              {
                  MPI_Recv(&mult,1,MPI_DOUBLE,j/rowsPerProc,...);
              }
              b[j] -= mult * b[i];
          }
     }
}
```

Figure 1: Code segment for Question 1(b). Each process holds `rowsPerProc` rows of the full matrix. The details of the initial matrix distribution, the broadcast of row i, and some arguments in the MPI function calls have been omitted for clarity.

Question 2

You are asked to numerically solve the following initial boundary value problem for $u(x,t)$ on the spatial domain $\Omega = [0,1]$, starting from a time $t = 0$,

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2} + ku\,.$$

You are provided with the initial condition $u(x,0)$ and the boundary conditions $u(0,t) = u_{\mathrm{L}}$ and $u(1,t) = u_{\mathrm{R}}$. You choose to approximate the solution on a regular mesh with $N+1$ nodes, where the nodes $i = 0$ and $i = N$ are boundary nodes. The time levels n are δt apart, so $t = n\delta t$.

(a) What is the spacing h between adjacent nodes on the mesh? **[1 mark]**

(b) Consider the case $v = 0$, so the term $-v\frac{\partial u}{\partial x}$ vanishes. Using central finite differences, show that the discrete version of the right hand side of the equation for an internal mesh node i is **[2 marks]**

$$D\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + ku_i\,.$$

(c) Using central differences for the term $-v\frac{\partial u}{\partial x}$ leads to unstable iteration, so it is instead suggested that you use upwinding. What is upwinding, and how would you apply it to this term for $v > 0$? **[2 marks]**

(d) The time derivative on the left hand side of the equation is approximated as

$$\frac{\partial u_i}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\delta t}\,.$$

This still leaves the choice of evaluating the right hand side at time t or at time $t + \delta t$. What are these two choices known as? Which is implicit? **[2 marks]**

(e) If the right hand side is evaluated at time $t + \delta t$, show that the finite difference equation (with $v > 0$) can be rearranged to give the following equation for u_i^{n+1} in terms of the solution vector at time level n, **[2 marks]**

$$u_i^{n+1} + \nu\left(u_i^{n+1} - u_{i-1}^{n+1}\right) - \mu\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) - \kappa u_i^{n+1} = u_i^n\,.$$

where $\nu = v\delta t/h$, $\mu = D\delta t/h^2$ and $\kappa = k\delta t$.

(f) The equation in part (e) can be written in matrix form as $A\mathbf{u}^{n+1} = \mathbf{b}(\mathbf{u}^n)$, where A is an $(N-1)\times(N-1)$ matrix and $\mathbf{b}$ is a constant $N-1$-vector. Write down three rows of the matrix A, and the corresponding elements of vector $\mathbf{b}$, far from the boundary nodes. **[4 marks]**

(g) The value of $u(x,t)$ at each boundary node is known. What is this type of boundary condition known as? You decide to absorb this boundary condition into the vector $\mathbf{b}$. Write down the finite difference equation for the node $i = 1$, and hence derive the first row of the matrix equation. **[4 marks]**

CONTINUED ON FACING PAGE

(h) Suppose you have applied to same procedure to the other boundary to produce the final matrix equation. What is the sparsity pattern for the matrix? What method would you use to solve the equation, and why? **[3 marks]**

[question 2 total: 20 marks]

Question 3

(a) (i) State one advantage and one disadvantage of using unstructured meshes rather than regular structured meshes. **[2 marks]**

(ii) Briefly describe the compressed sparse row format for the storage of a sparse matrix of size $N \times N$, with M non-zero elements. **[2 marks]**

(b) Delaunay triangulation is a method for constructing an unstructured mesh from a point cloud. It proceeds by bounding the points with a triangle, and adding new triangles for each point in turn, taking care to swap out edges that are illegal.

(i) What does it mean to say an edge shared between two mesh triangles is illegal, and how does edge swapping improve mesh quality? You may use a diagram if you like. **[2 marks]**

(ii) Figure 2(a) shows a point cloud with four points. The first bounding triangle is also provided. Carefully copy this diagram to your answer booklet and construct a mesh using Delaunay triangulation, incorporating each new point in the order specified in the figure. Provide your answer as a single diagram with the final mesh clearly marked. Also state for which points edge swaps were performed, if any. **[4 marks]**

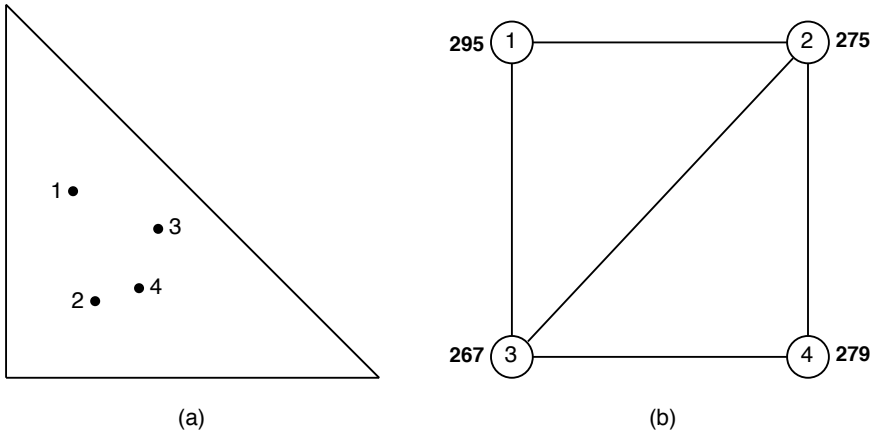

Figure 2: Diagrams for Question 3. (a) is for Question 3(b), where the numbers specify the order in which the points should be incorporated into the mesh. (b) is for Question 3(d), where the numbers in nodes refer to processes and the numbers adjacent to nodes to loads.

(c) (i) What are the two key requirements for a successful mesh partitioning, that is, one that achieves high parallel efficiency for the subsequent calculations? Justify your choices. **[2 marks]**

(ii) Two common algorithms for mesh partitioning are known as recursive coordinate bisection and recursive graph bisection. Choose one of these, stating your choice, and briefly describe how the method works. You may use diagrams if you wish. **[3 marks]**

CONTINUED ON FACING PAGE

(d) Unstructured meshes may be modified during the simulation in a process known as adaptive mesh refinement, and it is typical to perform data migration across the processes after each refinement phase. One such algorithm is known as the diffusion method, whereby a fraction c_{ij} of the load difference between each connected pair of nodes i and j is transferred at each iteration, rounding up. Consider the example given in Fig. 2(b), which shows a system partitioned into four processes with connected nodes specifying those with contacting domains.

(i) The coefficients c_{ij} for the load transfers are defined as

$$c_{ij} = \frac{1}{\max[\mathrm{degree}(i), \mathrm{degree}(j)] + 1} ,$$

where $\mathrm{degree(i)}$ is the degree of node i. Write down all of the c_{ij} for Fig. 2(b). **[1 mark]**

(ii) The numbers next to the nodes in Fig. 2(b) correspond to the load in each of the four processes after a round of mesh refinement. Implement the diffusion method until these loads are balanced. At each stage of the iteration provide a diagram showing the current load on each node, and the load to be transferred between nodes at the next iteration. What is the final balanced load? **[4 marks]**

[question 3 total: 20 marks]

[grand total: 60 marks]

END