EECS 739: Final Exam

Tuesday, May 12, 2015

Print Name and Signature_____

The rules for this exam are as follows:

- **Write your name on the front page of the exam booklet. Initial each of the remaining pages in the upper-right hand corner. Sign the front of the exam booklet. Your exam will not be graded if you have not signed the front page of the booklet.**
- The exam has 4 questions and 1 extra credit question. The exam is 10 pages long (including this page). Be sure you have all of the pages before beginning the exam.
- This exam will last for 150 minutes.
- Show **ALL** work for partial/full credit. This includes any definitions, mathematics, figures, etc.
- The exam is closed book and closed notes.
- You are allowed to use a calculator on the exam provided you use it to perform arithmetic computations only and show all of your work on the exam.
- No collaboration of any kind is allowed on the exam.

1. _____ (20 points)	4. _____ (40 points)
2. _____ (35 points)	EC. _____ (15 points)
3. _____ (35 points)	T. _____ (130 points)

1. (20 points; 2 points each) If possible, give an example of each of the following. If not possible, write "NOT POSSIBLE". **Do NOT include any additional justification.**

   (a) Three categories of partial differential equations.

   (b) A third-order accurate numerical PDE method in both space and time we discussed in class.

   (c) A parallel linear solver used in efficiently solving parabolic PDEs in parallel.

   (d) A numerical optimization method which has a cubic convergence rate.

   (e) A numerical optimization method which employs a line search.

   (f) A parallel numerical optimization method which does not involve solution of a linear system.

   (g) The two fundamentally different approaches to parallelizing code discussed in class.

   (h) The type of memory used by threads within a single block.

   (i) Two applications discussed in Prof. Zheng's guest lecture.

   (j) Two applications discussed in Prof. Shontz's research presentation on the last day of class.

2. (35 points) Consider the Du Fort-Frankel method for solving the heat equation (i.e., $u_t = u_{xx}$):

$$\frac{w_{i,j+1} - w_{i,j-1}}{2k} = \frac{w_{i+1,j} - w_{i,j+1} - w_{i,j-1} + w_{i-1,j}}{h^2}.$$

   (a) (15 points) Derive the Du Fort-Frankel finite-difference method for solving the heat equation.

   (b) (10 points) Write a parallel pseudocode for solving the above matrix system efficiently using MPI. **Be sure to specify the MPI commands that are used at each step.**

(c) (10 points) Specify the truncation error term for this finite-difference method. Unfortunately, this method has undesirable behavior. To see this, describe what happens to the truncation error when $h, k \to 0$ at the same rate. (It's OK to set $h = k$ for this analysis.)

3. (35 points)

(a) (10 points) Find and classify the critical points of $f(x, y) = 4 + x^3 + y^3 - 3xy$.

(b) (10 points) Perform one iteration of the steepest descent method on $f(x, y) = 4x^2 - 4xy + 2y^2$ using $(2, 3)$ as the starting point. **Use an exact line search for your calculations.**

(c) (15 points) Consider the following pseudocode for a line search which could be used with the steepest descent method:

```
// Line Search
// Try to get L<=Lmax so either f(xk+(L/2)*sk) < f(xk) or
// f(xk+L*sk) < f(xk).  Here xk is the current approximation to x.
L = Lmax;
Halvings = 0;
fL2 = f(xk+L*sk);
fL1 = f(xk+(L/2)*sk);
fLmax = fL2;
while ((fL1>=fk) & (fL2>=fk) & (Halvings<=20)) {
   L = L/2;
   Halvings = Halvings+1;
   fL2 = fL1;
   fL1 = f(xk+(L/2)*sk);;
}

// Find the quadratic interpolant
//             q(alpha) = a(1) + a(2) * alpha
//                 + a(3) * alpha^2
// of (0,fk), (L/2,fL1), (L,fL2).
//
Use Gaussian elimination to solve Ma = b, where
M = [1   0         0;
     1 (L/2) (L/2)^2;
     1   L       L^2]
and

b = [fk;
     fL1;
     fL2].

// Find the alpha that minimizes q on [0,Lmax].
Lcrit = -.5*a(2)/a(3);

if ((a(3)>0) & (0<=Lcrit) & (Lcrit<=Lmax)) {
   // There is a local minimum in the interval
   alpha = Lcrit;
}
else {
   // Check endpoints
   if fc < fLmax {
      alpha = 0;
   }
```

```
   else {
      alpha = Lmax;
   }
}
```

Consider using MPI to implement the steepest descent method using the above line search. Describe in words how additional processors could be used in an effective manner to improve upon a straightforward parallel implementation of the steepest descent method based on the above line search. Draw a picture to help explain your answer. **You do NOT need to write pseudocode for this problem.**

4. (40 points)

(a) (25 points) Recall that the matrix system for the Crank-Nicholson finite-difference method for use in solving the heat equation (with appropriate boundary conditions and initial conditions) is given by

$$Aw^{(j+1)} = Bw^{(j)}, j = 0, 1, 2, ...$$

where $\lambda = \alpha^2 k/h^2$,

$$w^{(j)} = \begin{pmatrix} w_{1j} \\ w_{2j} \\ \vdots \\ w_{m-1,j} \end{pmatrix},$$

$$A = \begin{pmatrix} 1+\lambda & -\lambda/2 & & \\ -\lambda/2 & \ddots & \ddots & \\ & \ddots & \ddots & -\lambda/2 \\ & & -\lambda/2 & 1+\lambda \end{pmatrix}, B = \begin{pmatrix} 1-\lambda & \lambda/2 & & \\ \lambda/2 & \ddots & \ddots & \\ & \ddots & \ddots & \lambda/2 \\ & & \lambda/2 & 1-\lambda \end{pmatrix}.$$

Write pseudocode for solving the relevant matrix system in parallel on a GPU. **Be sure to specify when blocks, threads, and kernels are being used.**

Hint: I recommend parallelizing the conjugate gradient method which is shown below.

Conjugate gradient method:

```
x_0 = initial guess
d_0 = r_0 = b-A*x_0;

while (norm(r_k) > tol) {
  alpha_k = ((r_k)^{T}*(r_k))/((d_k)^{T}*A*d_k);
  x_{k+1} = x_k + alpha_k * d_k;
  r_{k+1} = r_k - alpha*k * A * d_k;
  beta_k = ((r_{k+1}^{T})*(r_{k+1}))/((r_k)^{T}*r_k);
  d_{k+1} = r_{k+1} + beta_k + d_k;
}
```

(b) (15 points) Consider solving the heat equation using the above parallel method on a 1000 x 1000 mesh. How big is the resulting matrix system? Would you expect to obtain better performance using MPI or CUDA and why?

OPTIONAL: Extra-Credit Question

(15 points) Consider the PDE given by $xu_{xx} - u_{xy} + yu_{yy} + 3u_y = 1$ defined over $-\infty < x < \infty$ and $-\infty < y < \infty$.

(a) (10 points) Classify the above PDE as elliptic, parabolic, or hyperbolic. If the equation is of mixed type, identify the relevant regions and give the classification within each region.

(b) (5 points) Can a single parallel finite-difference method be used to solve the PDE for any values of x and y? Why or why not? **Explain your reasoning.**