CPS 5320 Theory-Based Qualifier Exam

09:00 am – 10:30 am on August 22, 2019

Name: _____

Student ID #: _____

Please read the following instructions carefully

1. This is a closed book exam.
2. The total time for this exam is 1 hour 30 minutes.
3. The exam is worth a total of 100 points.
4. You are permitted to use a simple (non-graphing, non-programmable) calculator. Cell phones, laptops, and all other web-enabled devices are not allowed.
5. Show sufficient work for full credit.

Number	Maximum Points	Earned Points
I	20	
II	20	
III	20	
IV	20	
V	20	
Total	100	

I. What application did we see which involved the "divergence" equation $U_x + V_y + W_z = 0$, and what physical property does this equation enforce? For an "axisymmetric" problem, $U = R(r,z)cos(\theta), V = R(r,z)sin(\theta), W = W(r,z)$, where $r = \sqrt{x^2+y^2}, \theta = tan^{-1}(y/x)$ are polar (or cylindrical) coordinates. Use the chain rule to convert the divergence equation to axisymmetric form, where only derivatives of R, W with respect to r, z appear.

II. Consider the elastic plate problem, $\nabla^2(\nabla^2 U) = q(x,y)$, with boundary conditions $M = 0, \frac{\partial M}{\partial n} = 0$, where $M = \nabla^2 U$ is the bending moment. Integrate both sides of the PDE and use the divergence theorem to find a condition on the load $q(x,y)$ which must be satisfied for this steady-state problem to have a solution. Why does this condition make sense physically? (Hint: what do the boundary conditions model?) Show that even if this condition is satisfied, the solution is not unique.

III. (a) A "call" option gives you the option to buy an asset at price E, at time T. What are the initial/final conditions and what are the boundary conditions, for the Black-Scholes equation for the value $V(s,t)$ of this option, where s is the price of the asset at time t.

(b) Same question, but for a "put" option, which gives you the option to sell the asset at price E, at time T.

IV. The damped membrane equation we studied was:

$$\rho u_{tt} + b u_t = T\nabla^2 u + f(x,y)$$

where ρ, b, T and $f(x,y)$ are the density (per unit area),frictional coefficient, tension and load. The total energy (kinetic plus potential) of the membrane is

$$E(t) = \iint_\Omega \left[\frac{1}{2}\rho u_t^2 + \frac{T}{2}\nabla u \cdot \nabla u - fu\right] \, dA$$

Show that if $u = g(x,y)$ on the boundary, the total energy is nonincreasing, and constant only if $b = 0$ or u has reached a steady-state. (Note that $f(x,y)$ and $g(x,y)$ are assumed to not be functions of time.)

V. The program PBACK below solves a nonsingular upper triangular system using back substitution, with the columns of U distributed cyclically over the processors, ie., 0,1,2,...,NPES-1,0,1,2,...,NPES-1,0,1,2,... Although each processor actually stores the entire matrix, it operates only on "its" columns. Modify PBACK (just mark your changes on the paper copy) so that the columns are distributed by "blocks," with the first NB=N/NPES columns on processor 0, the next NB columns on processor 1, etc: 0,0,0,...,0, 1,1,1,...,1,2,... You may assume that N will always be an integer multiple of NPES. This can be done by modifying only 2 lines. Hints: There is now a simpler way to specify the limits in loop 10.

```
      SUBROUTINE PBACK(U,X,B,N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C                               SOLVE U*X=B USING BACK SUBSTITUTION.
      DOUBLE PRECISION U(N,N),X(N),B(N)
      INCLUDE 'mpif.h'
C                               INITIALIZE MPI
      CALL MPI_INIT (IERR)
C                               NPES = NUMBER OF PROCESSORS
      CALL MPI_COMM_SIZE (MPI_COMM_WORLD,NPES,IERR)
C                               ITASK = MY PROCESSOR NUMBER (0,1,...,NPES-1).
C                               I WILL NEVER TOUCH ANY COLUMNS OF A EXCEPT
C                               MY COLUMNS
      CALL MPI_COMM_RANK (MPI_COMM_WORLD,ITASK,IERR)
      NB = N/NPES
      DO 20 I=N,1,-1
C                               I0 IS FIRST COLUMN >= I+1 THAT BELONGS
C                               TO ME
         L0 = (I+NPES-(ITASK+1))/NPES
         I0 = ITASK+1+L0*NPES
         SUMI = 0.0
         DO 10 J=I0,N,NPES
            SUMI = SUMI + U(I,J)*X(J)
   10    CONTINUE
         CALL MPI_ALLREDUCE(SUMI,SUM,1,MPI_DOUBLE_PRECISION,
     &    MPI_SUM,MPI_COMM_WORLD,IERR)
C                               JTASK IS PROCESSOR THAT OWNS U(I,I)
         JTASK = MOD(I-1,NPES)
C                               IF JTASK IS ME, CALCULATE X(I)
         IF (ITASK.EQ.JTASK) THEN
            X(I) = (B(I)-SUM)/U(I,I)
         ENDIF
C                               RECEIVE X(I) FROM PROCESSOR JTASK
         CALL MPI_BCAST(X(I),1,MPI_DOUBLE_PRECISION,JTASK,
     &   MPI_COMM_WORLD,IERR)
   20 CONTINUE
```

```
      CALL MPI_FINALIZE(IERR)
      RETURN
      END
```

Should the program run faster or slower now, or about the same, when $1 << NPES << N$? How much faster or slower (if any)? Explain.