Department of Scientific Computing

Written Preliminary Examination

Spring 2017

January 20–23, 2017

Instructions:

- Solve only 10 of the 11 questions as completely as you can.
- All questions are weighted equally.
- All parts of a question are weighted equally unless stated otherwise.
- If a particular question requests code, please email it (with your Student ID) to the person responsible for the question and Dr. Xiaoqiang Wang (wwang3@fsu.edu). The person responsible is listed at the beginning of each question.
- If you use web sources, please list them clearly.
- The exam is due back to David Amwake no later than 1:00 pm on Monday, January 23, 2017; no exceptions allowed.
- If you have any questions related to this exam as you work on it, please send an e-mail to the person responsible, *and* Dr. Xiaoqiang Wang (wwang3@fsu.edu).
- Write your Student ID on each of your answer sheets. Do not write your name on your answer sheets. When turning in your exam, include a cover page with your name and Student ID.

Q1. **Matrix** (Dr. Quaife)

Consider the function

$$f(\omega,\theta) = \frac{\cos\omega\cos\theta + \sin\omega\sin\theta - 1}{4(\cos\omega - \cos\theta)^2 + (\sin\omega - \sin\theta)^2}$$

(a) For a fixed θ, show that

$$\lim_{\omega\to\theta} f(\omega,\theta) = \frac{1}{2(3\cos^2\theta - 4)}.$$

(20 points)

(b) For a positive integer N, define the $N \times N$ matrix $A = (a_{ij})$ with entries

$$a_{ij} = f(\theta_i, \theta_j),$$

where $\theta_i = i\Delta\theta$, $i = 1, \ldots N$, and $\Delta\theta = 2\pi/N$. For the diagonal entries of A, use the limiting result from part (a). Write code that takes N as its input and outputs the matrix A. For $N = 128$, use your code to plot the 10^{th} and 50^{th} columns of A. Verify that they are both periodic and continuous at the diagonal entry. Use your code to also verify that A is symmetric. **(40 points)**

(c) Consider partitioning the matrix A into the hierarchical block structure illustrated in Figure 1. Each block comes from dividing its parent block in four equal parts. For the three values $N = 2048, 4096, 8192$, write the number of singular values greater than 10^{-12} of each childless block from Figure 1. Discuss the behavior of the ranks of the blocks with respect to N. **(40 points)**

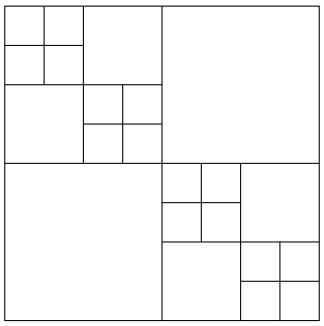

Figure 1

Q2. **Partial Differential Equations** (Dr. Quaife)

Consider the one dimensional heat equation with periodic boundary conditions

$$\begin{aligned} u_t &= u_{xx} && x \in [0, 2\pi],\ t > 0 \\ u(0,t) &= u(2\pi,t), && t > 0, \\ u_x(0,t) &= u_x(2\pi,t), && t > 0, \\ u(x,0) &= u_0(x), && x \in [0, 2\pi]. \end{aligned}$$

(a) Consider discretizing the one dimensional heat equation in time, but not in space. Perform the time discretization with a single time step from t^m to t^{m+1}. What is the boundary value problem that must be solved if Backward Euler is used? What is the boundary value problem that must be solved if Crank-Nicolson is used? **(15 points)**

(b) Write the Fourier coefficients of $u(x, t^{m+1})$ in terms of the Fourier coefficients of $u(x, t^m)$ for both Backward Euler and Crank-Nicolson. In both cases, your answer should be of the form

$$\hat{u}_k^{m+1} = \Lambda(k, \Delta t)\hat{u}_k^m, \quad k \in \mathbb{Z},$$

where $\hat{u}_k^m$ are the Fourier coefficients of $u(x, t^m)$, and $\hat{u}_k^{m+1}$ are the Fourier coefficients of $u(x, t^{m+1})$. **(15 points)**

(c) We know that the solution of the heat equation smooths as a function of time. Given the functions Λ from part (b), discuss the smoothing properties of Backward Euler and Crank-Nicolson. Discuss, in particular, the smoothing of high frequency components (large k). **(15 points)**

(d) Implement both methods derived in part (b). Discretize the spatial domain with $N = 1024$ points. Use a time domain of $[0, 1]$ and the initial condition $u(x, 0) = e^{\cos x}$. Perform a convergence study by tabulating the errors for various time step sizes Δt. For an exact solution, use the numerical solution formed with 10,000 time steps of the respective method. What order of convergence do you observe? Is this what you expected? Explain. **(35 points)**

(e) Use your code from part (d) with $N = 1024$, $\Delta t = 0.01$, and the initial condition

$$u_0(x) = \begin{cases} 1 & x \in (\frac{\pi}{2}, \frac{3\pi}{2}) \\ 0 & \text{otherwise.} \end{cases}$$

Describe any qualitative differences you observe between Crank-Nicolson and Backward Euler. You may include plots. Relate your observations to your discussion in part (c). **(20 points)**

Q3. **Linear Algebra**: (Dr. Peterson)

Assume we have a singular value decomposition (SVD) of the real $m \times n$ matrix A where $m \geq n$ given by $A = U\Sigma V^T$ with singular values $\sigma_j \geq 0$. Assume for simplicity that A is full rank.

1. Show that the nonzero singular values of A are the square roots of the nonzero eigenvalues of $A^T A$. Why is finding the eigenvalues of $A^T A$ not a good strategy for computing the singular values of A? **(20 points)**

2. Explain how the SVD of A can be used to compute a rank $k < n$ approximation to A. **(20 points)**

3. If we want a rank $k < n$ approximation to A, we have to choose k. If we choose a large value of k, then of course we get a better approximation but we are doing more work. On the other hand, if we choose too small a value of k then we don't have a very good approximation. Decide what strategy you would use to decide on an approximation of A in order to balance work and accuracy. Apply your strategy to the matrix

$$\begin{pmatrix} 4 & 5 & 4 & 7 & 5 & 7 & 4 \\ 6 & 8 & 5 & 8 & 8 & 11 & 6 \\ 5 & 7 & 5 & 9 & 9 & 10 & 7 \\ 6 & 8 & 5 & 8 & 9 & 11 & 7 \\ 4 & 3 & 2 & 3 & 4 & 7 & 3 \\ 4 & 6 & 5 & 8 & 8 & 8 & 6 \\ 1 & 3 & 3 & 5 & 5 & 5 & 4 \\ 5 & 6 & 6 & 8 & 8 & 9 & 7 \end{pmatrix}.$$

   (30 points)

4. For simplicity, assume that A is $m \times m$ for this part. Define the matrix H as

$$H = \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}.$$

   Show that the matrix

$$Q = \begin{pmatrix} V & V \\ U & -U \end{pmatrix}$$

   diagonalizes H. How are the singular values of A related to the eigenvalues of H? Hint: Consider HQ. The way that the SVD is computed in practice is related to this idea. **(30 points)**

Q4. **Ordinary Differential Equations** (Dr. Meyer-Baese)

Consider a predator-prey model

$$\begin{aligned} \frac{dx}{dt} &= x\left(r - r\frac{x}{K} - ay\right), \quad r, K, a > 0 \\ \frac{dy}{dt} &= y(-b + cx), \quad b, c > 0 \end{aligned} \tag{1}$$

The variable x represents the density of the prey species and the variable y the predator. It is assumed, in the presence of the predator, that the prey grows logistically. Also, in the absence of the prey, the predator dies out (exponentially). The term ay represents the per capita loss of prey to the predator and cx represents the per capita gain to the predator.

(a) Determine the equilibria of the predator prey model. **(20 points)**

(b) Determine the Jacobian matrix of this system. **(30 points)**

(c) Determine the eigenvalues of the Jacobian at all equilibria. **(30 points)**

(d) Use RK45 to solve (a), and report the amount of time required to solve it. You may use a cannned routine, and set the relative tolerance to 10^{-6}. **(20 points)**

Q5. **Interpolation** (Dr. Shanbhag)

Lagrange polynomial interpolation yields a polnomial $p(x)$, passing through the $n+1$ points (x_0, f_0), (x_1, f_1), ..., (x_n, f_n):

$$p(x) = \sum_{i=0}^{n} L_i(x) f_i, \text{ with } L_i(x) = \prod_{j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{2}$$

Two commonly cited disadvantages of Lagrange interpolation, compared to Newton's divided differences, include: (i) evaluation of $p(x)$ at each point x costs $\mathcal{O}(n^2)$,[1] and (ii) adding a new data point (x_{n+1}, f_{n+1}) requires recomputing the polynomial from scratch. Let us consider rewriting the formula 2 to partially mitigate these problems.

Define $L(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$, and

$$w_i = \prod_{j \neq i}^{n} \frac{1}{x_i - x_j}, \quad i = 0, 1, ..., n.$$

1. Show that the Lagrange interpolation formula (eqn. 2) can be rewritten as:

$$p(x) = L(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i} f_i \tag{3}$$

   (10 points)

2. Show that the asymptotic cost of assembling all the w_i is still $\mathcal{O}(n^2)$. Note that this cost is incurred only once, as w_i do not depend on x. **(10 points)**

3. Given all the w_i, build an $\mathcal{O}(n)$ algorithm to evaluate $p(x)$ at any x using eqn. 3. **(15 points)**

4. How would you incorporate a new data point (x_{n+1}, f_{n+1}) without starting from scratch. What is the asymptotic cost of this method? **(40 points)**

5. Consider the function,

$$f(x) = |x| + \frac{x}{2} - x^2, \quad x \in [-1, 1].$$

   Use $n+1 = 101$ Chebyshev nodes (x_i) to find the polynomial $p_{100}(x)$ that interpolates $(x_i, f(x_i))$. Plot $p_{100}(x)$ and $f(x)$ together. **(15 points)**

6. Consider interpolating a different function $g(x)$, which shares the same domain $x \in [-1, 1]$, using the same 101 nodes x_i as above. What advantage does eqn 3 have over Newton's divided differences? **(10 points)**

[1] in divided differences, building the table costs $\mathcal{O}(n^2)$, but subsequent evaluations $p(x)$ can be performed in $\mathcal{O}(n)$ using Horner's method, or some other variant.

Q6. **Numerical Differentiation** (Dr. Shanbhag)

Consider the derivative of the function,

$$f(x) = \frac{e^x}{\sqrt{\sin^3 x + \cos^3 x}},$$

at $x^* = 1.5$. The true derivative is $f_t'(1.5) = 4.0534278938986206933912 86$.

Two numerical approximations for the first derivative are given by:

$$\begin{aligned} f_c'(x) &\approx \frac{f(x+h) - f(x-h)}{2h} && \text{centered difference formula} \\ f_i'(x) &\approx \frac{\text{Im}(f(x+ih))}{h} && \text{complex step derivative} \end{aligned} \tag{4}$$

where $i = \sqrt{-1}$, $\text{Im}(x)$ denotes the imaginary part of x, and h is the step size.

1. Perform Taylor series expansions of $f(x+h)$, $f(x-h)$, and $f(x+ih)$, and show that the leading truncation error term for both formulas in eqn. 4 is $\mathcal{O}(h^2)$. **(20 points)**
2. Vary h between 10^{-14} to 10^{-2} and plot the absolute errors $\epsilon_c(h) = |f_c'(1.5) - f_t'(1.5)|$ and $\epsilon_i(h) = |f_i'(1.5) - f_t'(1.5)|$ as a function of h. **(60 points)**
3. What is the minimum error ϵ_c and ϵ_i, and the optimal step size h^* for both methods? You may read off approximate values from your plot above. **(10 points)**
4. Why does the complex step derivative not suffer from roundoff error like the centered difference formula? **(10 points)**

Q7. **Quadrature** (Dr. Shanbhag)

Consider the integral,

$$I = \int_0^1 dx \exp(10x^2) \int_{-\infty}^{x} dy \exp(-10y^2). \tag{5}$$

Use Gauss-Laguerre to evaluate the inner integral, and Gauss-Legendre to determine the outer integral. Adjust the number of grid points to estimate I within ± 0.1 of the true value.

Q8. **Optimization** (Dr. Navon)

Part A

Consider the constrained minimization problem: Minimize $f(x) = \frac{1}{2}[x_1^2 + \frac{1}{3}x_2^2]$ subject to $x_1 + x_2 = 1$. Use an unconstrained minimization method such as the conjugate gradient algorithm or the BFGS Q-N one to write a small code in order to compare method of Augmented Lagrangian with that of quadratic penalty where the augmented Lagrangian function is given by

$$L(x, c_k, \lambda_k) = \frac{1}{2}[x_1^2 + \frac{1}{3}x_2^2] + \lambda_k(x_1 + x_2 - 1) + \frac{1}{2}c_k((x_1 + x_2 - 1)^2. \tag{6}$$

In the quadratic penalty method $\lambda_k = 0$ for all k. Use the following updating of penalty coefficient and Lagrange multiplier in the Augmented Lagrangian for the multipliers:

$$\lambda_{k+1} = \lambda_k + c_k(x_1^{(k)} + x_2^{(k)} - 1)d \tag{7}$$

and $\lambda_0 = 0$. For the penalty coefficients try two options:

$$c_{k+1} = 10c_k, \quad c_0 = 0.2$$

or try

$$c_k = 0.1 \times 2^k; \quad c_k = 0.1 \times 4^k; \quad c_k = 0.1 \times 8^k.$$

Put the computational results in a form of table for $k, x_1^{(k)}, x_2^{(k)}$ for about 10-15 iterations for both methods and different update variants.

Using first order optimality conditions minimization of $L(x, c_k, \lambda_k)$ yields:

$$x_1^{(k)} = \frac{c_k - \lambda_k}{1 + 4c_k}, \quad x_2^{(k)} = \frac{3(c_k - \lambda_k)}{1 + 4c_k}.$$

So it is evident that the optimal solution is:

$$x^* = (0.25, 0.75), \quad \lambda^* = -0.25.$$

In order to start your calculations use as initial guess point: $x_0 = (0.11, 0.33)$. Your results should show (if your coding is correct) that method of augmented Lagrangian requires a smaller number of minimizations to obtain the solution than quadratic penalty and that the number of minimizations for both penalty and A-L methods decreases when the penalty parameter is increased at a faster rate.

However the effects of ill-conditioning are felt more under these circumstances when the unconstrained minimization is carried out numerically.

Part B

We consider an approach known as the method of multipliers or the augmented Lagrangian method. This algorithm is related to the quadratic penalty algorithm but it reduces the possibility of ill conditioning by introducing explicit Lagrange multiplier estimates into the

function to be minimized, which is known as the augmented Lagrangian function

$$L_A(x, \lambda; \mu) \doteq f(x) - \sum_{i \in \epsilon} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \epsilon} c_i^2(x). \tag{8}$$

Consider

$$\min_x f(x) \quad \text{subject to} \quad c_i(x) = 0, \quad i \in \mathcal{E} \tag{9}$$

The pseudo-code for (9) is:

Algorithm 1 Augmented Lagrangian Method-Equality Constraints

Given $\mu_0 > 0$, tolerance $\tau_0 > 0$, starting points x_0^s and λ^0;
for $k = 0, 1, 2, ...$
    Find an approximate minimizer x_k of $L_A(\cdot, \lambda^k; \mu_k)$, starting at x_k^s,
        and terminating when $\|\nabla_x L_A(x_k, \lambda^k; \mu_k)\| \leq \tau_k$;
    if a convergence test for (9) is satisfied
        stop with approximate solution x_k;
    end (if)
    Update Lagrange multipliers using the following equation to obtain λ^{k+1};

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_k), \text{for all } i \in \mathcal{E}. \tag{10}$$

    Choose new penalty parameter $\mu_{k+1} \geq \mu_k$;
    Set starting point for the next iteration to $x_{k+1}^s = x_k$;
    Select tolerance τ_{k+1};
end(for)

Now consider the code ALGENCAN (please download it from `https://people.sc.fsu.edu/~wwang3/algencan-3.1.0.zip`) that solves equality constrained problems using the Augmented Lagrangian method. ALGENCAN is a FORTRAN code for general nonlinear programming that does not use matrix manipulations at all and, so, is able to solve extremely large problems with moderate computer time. The general algorithm is of Augmented Lagrangian type and the sub problems are solved using GENCAN. GENCAN (included in ALGENCAN) is a FORTRAN code for minimizing a smooth function with a potentially large number of variables and box-constraints.
Please use ALGENCAN (using the instructions in the README page) to solve the toy problem 'toyprob' as an example to code and solve your own problem, as well as the other simple examples 'toyprob2', and 'toyprob3'. The toy problems (written in c or FORTRAN) can be found in: algencan-3.1.0\sources\examples\.
As a final task please flowchart the code in ALGENCAN and find which routines correspond to the Algorithm 1 (Augmented-Lagrangian Framework pseudocode).

Q9. **Parallel Computing** (Dr. Huang)

Pass	Processor 0 $E_{proc,0}$	Processor 1 $E_{proc,1}$	Processor 2 $E_{proc,2}$	Processor 3 $E_{proc,3}$
0	0 1 0 1 0-1	2 3 2 3 2-3	4 5 4 5 4-5	6 7 6 7 6-7
1	0 1 6 7 0-6, 0-7 1-6, 1-7	2 3 0 1 2-0, 2-1 3-0, 3-1	4 5 2 3 4-2, 4-3 5-2, 5-3	6 7 4 5 6-4, 6-5 7-4, 7-5
2	0 1 4 5 0-4, 0-5 1-5, 1-4	2 3 6 7 2-6, 3-6 2-7, 3-7	4 5 0 1	6 7 2 3

Figure 2: Parallel algorithm for computing the total energy of 8 particles with 4 processors. The interaction energies to be computed on each processor at each pass is listed under the boxes.

A major task in molecular dynamics is to compute the total system's energy. We consider the interaction between particles to be pairwise, and assume that no atom can interact with itself. The sequential code for computing the total energy is

```
energy = 0.0;
for (i=0;i<n;i++)
       for (j=i+1;j<n;j++)
           energy = energy + calculate_energy(i,j);
```

The function **calculate_energy(i,j)** computes the energy between atoms i and j. The computational cost of the above code is $n(n-1)/2$. For a system with a large number of atoms, we need to parallelize the code over processors. A possible way is to distribute the n atoms uniformly among p processors, assuming that n is divisible by p. Each processor has n/p atoms, and the atoms are then passed from one processor to another in a ring-like fashion. That is, in each pass, the processor i receives n/p atoms from the processor $i-1$. The algorithm is described in the figure. Consider that we have 8 atoms and 4 processors. Initially, we assign particles 0 and 1 to Processor 1, particles 2 and 3 to Processor 2, particles 4 and 5 to Processor 3, and particles 6 and 7 to Processor 4. The assignment is denoted by the red numbers in the upper (gray) boxes. The particles in the gray boxes are called

self-particles. For the processor i, its lower box shows the guest particles that are from the processor $i-1$ at each pass. In each box, the interaction energies between the guest- and self-particles are computed and are accumulated in a local variable of that processor. For instance, at the Pass #1, the processor 2 received the particles 2 and 3 from the processor 1. The interaction energies 4-2, 4-3, 5-2, and 5-3 are computed and are added to the local variable $E_{proc,2}$ of the processor 2.

We only need to perform $p/2$ passes. In the last pass (the Pass #2), we do not need to compute the interaction energies on the Processors 2 and 3. In the end, all processors send their $E_{proc,j}$ to the Processor 0 which sums up all interactions to obtain the total energy. Define the interaction energy e_{ij} between particles i and j as $e_{ij} = \sqrt{i+j}$. The total energy is $E = \frac{1}{2}\sum_i \sum_{j \neq i} e_{ij}$. Write an MPI program to realize the calculations described in the figure. For the purpose of simplicity, you do not need to consider the general case (n particles distributed over p processors), but only consider the case of 8 particles distributed over 4 processors. Submit your code. Print $E_{proc,j}$ of each processor at each pass. Print the guest atoms on each processor at each pass. Compare your result to that obtained using the sequential code.

Q10. **Data Structure** (Dr. Beerli)

Assume that you have n 'species' (for example $n = 5$: toad, sea bass, bird, chimpanzee, and cow; see figure). You do not need to be a biologist to answer this question, but remember all species are leaves on the tree ,and they are connected by a non-reticulate graph through internal nodes. Usually, we assume that each internal node has one ancestor and two dependents. One can write these trees in a condensed format (Newick tree-format: you can find more information here: https://en.wikipedia.org/wiki/Newick_format and here: http://evolution.genetics.washington.edu/phylip/newicktree.html).

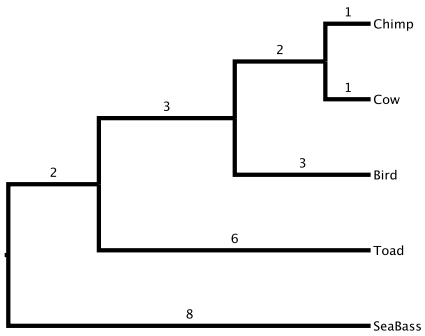

Figure 3: Example of a phylogenetic tree

This tree has the following NEWICK string:

```
((((Chimp:1.0,Cow:1.0):2,Bird:3):3,Toad:6):2,SeaBass:8);
```

Furthermore, each species has two variables associated with it: *name*, *number of offspring*. On average Chimpanzees have 1 offspring, Cows have 1.5 offspring, Birds have 5 eggs, Toads have 400 eggs, Seabass have 200,000 eggs.

1. Write a code in python or matlab, that reads a newick string from a file (use the one I supplied above), I expect that the code could read another NEWICK string just fine, thus do not hardcode this particular tree. Most commonly this task is done recursively, see Algorithm 5 in https://people.sc.fsu.edu/~pbeerli/isc5317-notes/pdfs/09-trees.pdf, or non-recursively (algorithm 6). Pick either recursive or non-recursive. The PDF has more general information about these trees and how to manipulate them. No output is needed for this section, but it is the prerequisite for the next tasks. **(50 points)**

2. Write a preorder traversal that finds the *name* of the species and associates the number of offspring with it. You can either read the name offspringnumber pair from a file or hardcode them into an array in your program. **(25 points)**

3. Write a postorder traversal that prints out the names of the dependents and the average of the number of offspring. The output could look likes this

   ```
   Chimp_Cow 1.25
   Chimp_Cow_Bird 3.125
   ```

```
Chimp_Cow_Bird_Toad 201.5625
Chimp_Cow_Bird_Toad_SeaBass 100,100.78125
```

(25 points)

4. Submit the code for all three tasks, only the last task will produce an output.

Q11. **Statistics** (Dr. Beerli)

In a coin tossing experiment using a quarter you were allowed to toss the coin only ONCE. You get 'Head'. The task at hand is to find out whether the coin is cheated or whether the coin is fair. The parameter of interest is the probability of head p.

1. Maximum likelihood
    - What is the likelihood function Prob($'Head'|n, k, p$) where n is the total number of tosses, k is the number of Heads, and p is the probability of Head? **(10 points)**
    - What is the maximum likelihood estimate? **(10 points)**
    - Show a graph of the likelihood curve. **(10 points)**
    - Based on the likelihood, is the coin rigged? Discuss your answer. **(10 points)**
2. Bayesian Inference
    - What is the formula of the posterior Prob($p|'Head', n, k$) where n is the total number of tosses, k is the number of Heads, and p is the probability of Head. (you can ignore the denominator in Bayes rule) **(10 points)**
    - Use a Beta distribution for the prior. The Beta distribution has two parameters, discuss two different, but sensible parameter settings. **(10 points)**
    - Show a graph of the posterior. **(10 points)**
    - Based on the Bayesian analysis, is the coin rigged? Discuss your answer. **(10 points)**
3. Which of the two methods would you use, if this would be your own analysis, explain! **(25 points)**
