January 2015 Written Preliminary Exam

Department of Scientific Computing

Exam is due back to Mark Howard no later than 12 noon on Monday, January 12.

Instructions:

- Do as many problems as completely as you can but NO MORE than ten problems.
- All parts of a problem are weighted equally unless otherwise stated at the end of the problem.
- Do NOT write you name on your answer sheets. Rather write your Student ID number on each page.
- Write legibly. If we can not read your writing, we can't grade it. You do not have to use Latex for your answers.
- If you have any questions related to a problem email the faculty member responsible for the question (listed on each problem) and Prof. Peterson (jpeterson@fsu.edu).
- Codes used to solve a question should be emailed to the person responsible for the question and Prof. Peterson.
- You can use any outside sources (books, web, etc.) but these should be documented at the end of each problem. You may not discuss the exam with any individual other than the faculty members responsible for the exam before noon on January 12.

1. **Numerical Linear Algebra** (Peterson)

Let A be an $m \times n$ real matrix and assume that $m > n$ and A is full rank. In this case, we can define an analog of the inverse of a square matrix for rectangular matrices. We denote this generalization as $A^\dagger$ and define it as

$$A^\dagger = (A^T A)^{-1} A^T \,. \tag{1}$$

In this problem we will need the specific rectangular matrix

$$A_1 = \begin{pmatrix} 2 & 10.2001 \\ 1 & 5.1001 \\ 5 & 25.4999 \end{pmatrix}$$

In the numerical calculations below give your answers to 8 digits of accuracy.

a.

(i) Prove that $A^\dagger A = I_{n\times n}$ where $I_{n\times n}$ denotes the $n \times n$ identity.

(ii) Let B be an $m \times m$ matrix with orthonormal columns. Show that

$$(BA)^\dagger = A^\dagger B^T \,.$$

b. A naive approach to calculating $A^\dagger$ is to use the formula (1) directly. Calculate $A_1^\dagger$ using this approach either by hand or with the aid of Matlab. Explain why this approach is not recommended.

c.

(i) Show how the QR decomposition of a matrix can be used to find $A^\dagger$. For uniformity, use the notation $A = QR$ for the QR decomposition and let R_1 denote the nonsingular, upper triangular portion of R. Give your answer in terms of of Q and R_1.

(ii) Use your formula in (i) to compute $A_1^\dagger$. You can use Matlab's built-in command to get the QR decomposition but provide your output; don't just give the answer.

d.

(i) Show how the SVD decomposition of a matrix can be used to find $A^\dagger$. For uniformity, use the notation $A = U\Sigma V^T$ for the SVD decomposition and let D denote the square nonzero portion of Σ; i.e.,

$$\Sigma = \begin{pmatrix} D \\ 0 \end{pmatrix}$$

where here 0 represents the appropriately sized zero matrix. Write your answer in terms of U, V, Σ and/or D.

(ii) Use your formula to compute $A_1^\dagger$. You can use Matlab's built-in command to get the SVD decomposition but provide your output; don't just give the answer.

e. What application have you encountered where $A^\dagger$ needs to be computed? Explain.

(a) - 20%; (b) - 10%; , (a) - 30%; , (a) - 30%; (e) - 10%

2. **Numerical Linear Algebra** (Peterson)

Assume that we want to solve the two-point boundary value problem (BVP)

$$-u''(x) = f(x), \quad 0 < x < 1 \tag{2}$$

with homogeneous Dirichlet boundary conditions. If we discretize the domain using a uniform grid $x_0 = 0$, $x_i = x_{i-1} + h$, for $i = 1, \ldots, m+1$ where $h = 1/(m+1)$ and use a second centered difference quotient to approximate the second derivative then it is well known that the coefficient matrix is an $m \times m$ tridiagonal matrix with $2/h^2$ on the main diagonal and $-1/h^2$ on the sub- and super-diagonals. In this problem this symmetric tridiagonal matrix is denoted A.

a. Show that A is positive definite by first demonstrating that

$$\vec{u}^T A \vec{u} = \frac{1}{h^2}\left[u_1^2 + u_m^2 + \sum_{i=2}^{m-1}(u_i - u_{i-1})^2\right]$$

for any $\vec{u} = (u_1, u_2, \ldots, u_m)$ in $\mathbf{R}^m$. Explain how this expression guarantees that A is positive definite.

b. A stationary iterative method for the linear system $B\vec{x} = \vec{f}$ has the general form $\vec{x}^{k+1} = P\vec{x}^k + \vec{c}$.

(i) Assume that we can know that

$$\lim_{k\to\infty} \|P^k\|_\alpha = 0_{n\times n} \tag{3}$$

where $\|\cdot\|_\alpha$ is a matrix norm obtained from a vector norm by

$$\|B\|_\alpha = \sup_{\vec{x}\neq 0} \frac{\|B\vec{x}\|}{\|\vec{x}\|}$$

and $0_{n\times n}$ denotes the $n \times n$ null matrix. Prove that the iterative method converges, i.e., that

$$\lim_{k\to\infty} \|\vec{x}^k - \vec{x}\| = 0$$

where $\vec{x}$ is the exact solution to the linear system $B\vec{x} = \vec{f}$.

(ii) The Jacobi method is probably the simplest stationary iterative method. Derive P and $\vec{c}$ for the Jacobi method when applied to a general problem $B\vec{x} = \vec{f}$ by using the splitting $B = M - N$ for appropriately chosen M, N. Explicitly give P for our tridiagonal matrix A.

c. The following lemma can be proved concerning the convergence of a stationary iterative method. You do NOT have to prove it.

Lemma. Let B be a symmetric, positive definite matrix where we have the splitting $B = M - N$ with M invertible. If the matrix $M^T + N$ is positive definite then $\rho(M^{-1}N) < 1$ where $\rho(Q)$ denotes the spectral radius of Q.

(i) If the Jacobi method is applied to our matrix A generated from discretizing the two-point BVP (2), use this lemma to guarantee that the Jacobi method will converge.

(ii) The following two matrices satisfy the hypotheses of this Lemma. Which matrix do you think will converge faster when the Jacobi method is applied? Why?

$$B_1 = \begin{pmatrix} 5 & -1 & -1 \\ -1 & 5 & -1 \\ -1 & -1 & 5 \end{pmatrix} \qquad B_2 = \begin{pmatrix} 8 & -2 & -2 \\ -2 & 8 & -2 \\ -2 & -2 & 8 \end{pmatrix}$$

d. What numerical method would you choose to use to efficiently calculate the spectral radius of the iteration matrix for the Jacobi method when applied to our specific matrix A? Justify why you chose to use this algorithm. Implement your method for $m = 100$ and iterate until you have the eigenvalue (computed using the Rayleigh quotient) to three significant digits. Provide your code and an output file that gives the iteration number, the approximate eigenvector and eigenvalue.

3. **Approximation Theory** (Gunzburger)

a. The best $L^\infty[0,1]$ linear polynomial approximation $p_1^*(x)$ of the function $f(x) = \sqrt{1+x^2}$ on $[0,1]$ is given by

$$p_1^*(x) = \frac{2c+1}{2} + (\sqrt{2}-1)x \quad \text{where} \quad c = \sqrt{\frac{1}{2}\sqrt{2}-1}.$$

Determine the error $\|f(x) - p_1^*(x)\|_{L^\infty[0,1]}$ in the best approximation $p_1^*(x)$.

b. Determine the best $L^2(0,1)$ linear polynomial approximation of $f(x) = \sqrt{1+x^2}$ on $[0,1]$.

c. Determine the $L^\infty[0,1]$ norm of the least-squares approximation found in part b.

4. **Numerical ODEs** (Erlebacher)

Consider the system of equations

$$dx/dt = -80.6x + 119.4y + \epsilon\cos(3t), \quad 0 < t \leq 5 \tag{4}$$
$$dy/dt = 79.6x - 120.4y, \quad 0 < t \leq 5 \tag{5}$$

with initial conditions $x(0) = 1$ and $y(0) = 4$, and $\epsilon = 0.1$.

a.

(i) Write down an explicit one-step third order method to solve a single initial value problem (IVP). You do not have to verify that it is third order. Then apply this method to the given system of IVPs.

(ii) Implement your algorithm and and plot your solutions for t in the range [0,5]. You may choose Δt.

(iii) Theoretically derive the maximum allowable time step to avoid instabilities. Provide evidence that the time step found is correct via numerical simulation by plotting the solution for a time step slightly above and slightly below the theoretical limit.

(iv) Change the initial conditions to avoid the small time step found in part (ii). What is the associated maximum time step with these new initial conditions?

b.

(i) Derive an IVP whose solution is

$$x(t) = e^{-t} + e - 4e^{-10^4 t} \tag{6}$$

(ii) Solve the problem numerically using a second order method (your choice) and the third order method you chose in (a) for the range $[0, 1]$. Starting with a time step Δt that is stable, compute the solution at a sequence of time steps Δt, $(\Delta t)/2$, $(\Delta t)/2^2, \ldots$. For each method make a table of your results which includes the time step, the approximation at $t = 1$, the error, the numerical rate of convergence and the number of time steps required. Plot the error for each method at $t = 1$ in such a way that your numerical rates are demonstrated.

(ii) Is the solution smooth? Why is the time step so small? What numerical method would you use to avoid the very small time steps without changing the initial conditions?

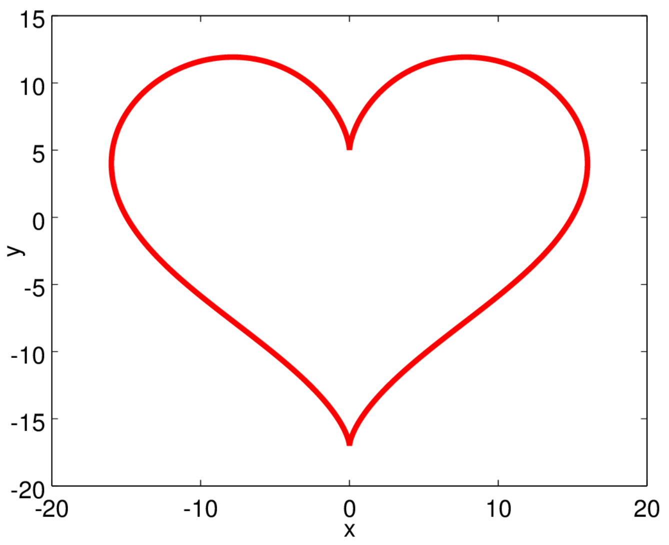
5. Numerical Quadrature (Shanbhag)

The radius of gyration R_G of an object is the average distance of points from its center-of-mass, $\mathbf{r}_{cm}$.

Consider a heart-shaped domain $\mathcal{D} \in \mathbf{R}^2$ enclosed by the curve:

$$\mathbf{h}(t) = \left(16\sin^3 t,\ 13\cos t - 5\cos 2t - 2\cos 3t - \cos 4t\right), \quad 0 \le t < 2\pi.$$

Let $\mathbf{r} = (x, y)$ denote a point in $\mathbf{R}^2$



a. What numerical method do you think is appropriate to use to estimate the area of this heart shaped region? Explain your choice.

b. Use your method from (a) to numerically estimate each of the following integrals to two digits of accuracy.

(i) The area of $\mathcal{D}$, given by,

$$A = \int_{\mathbf{r}\in\mathcal{D}} dx\ dy.$$

(ii)The center of mass,

$$\mathbf{r}_{cm} = \frac{1}{A}\int_{\mathbf{r}\in\mathcal{D}} \mathbf{r}\ dx\ dy.$$

(iii)The radius of gyration,

$$R_G^2 = \frac{1}{A}\int_{\mathbf{r}\in\mathcal{D}} (\mathbf{r} - \mathbf{r}_{cm})^2\ dx\ dy,$$

where $(\mathbf{r} - \mathbf{r}_{cm})^2$ is the square of the Euclidean distance between $\mathbf{r}$ and the center of mass.

c. Comment on the amount of additional work required to obtain three digits of accuracy.

6. **Statistics** (Beerli)

Assume that you have the following dataset:

$$\{28.6864, 26.4997, 21.9809, 32.2147, 6.20551, 1.57577, 26.789, 13.7372, 15.0708, 8.15355\}$$

a. Develop a Bayesian inference strategy that estimates a mean μ and a standard deviation σ of a normal distribution. You will need to give formulae for the prior distribution, the likelihood, and give results for the posterior mode for μ and σ.

b. Does the prior choice matter? Use a different prior (show the formula) and show the posterior mode for both parameters. In a short paragraph discuss the difference between the runs with the different priors.

7. **Fourier Analysis** (Meyer-Baese)

a. Assume that you are given a periodic signal $x(t)$ expressed by the following Fourier series:

$$x(t) = 12\cos(8t) + 4\cos(5t - \frac{\pi}{2}) - 8\sin(10t - \frac{\pi}{3})\,.$$

(i) Sketch the exponential Fourier series spectra.

(ii) Write the exponential Fourier series for $x(t)$.

b. Using the definition of the Fourier Transform

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt \quad \Longleftrightarrow \quad x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\,d\omega$$

compute the Fourier Transform $X(\omega)$ for

$$x(t) = \begin{cases} e^{-at} & t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

8. **Optimization** (Wang)

Consider a penalty method with quadratic penalty type

$$\min \ f(x_1, x_2) = -x_1 x_2$$

such that $g(x) = x_1 + 2x_2 - 4 = 0$. That is, we want to minimize

$$\min \ \mathcal{P}(x, \rho) = -x_1 x_2 + \frac{1}{2}\rho(x_1 + 2x_2 - 4)^2$$

where ρ denotes the penalty parameter.

a. Use necessary conditions of unconstrained minimization problem to show that for $\rho > \frac{1}{4}$,

$$x_1(\rho) = x_1 = \frac{8\ \rho}{4\ \rho - 1}$$

$$x_2(\rho) = x_2 = \frac{4\ \rho}{4\ \rho - 1}$$

and $x^{\text{opt}} = (2, 1)^T$.

b. Compute the condition number of the Hessian $\nabla_x^2 \mathcal{P}(x, \rho)$ at $X(\rho)$ and show that it is approximately equal to $\frac{25\,\rho}{4}$.

c. On the basis of this result, comment on the ill-conditioning of the penalty method when $\rho \to \infty$.

9. **Finite Element Method** (Burkardt)

Assume that we want to solve the modified heat equation in one spatial dimension for $u = u(x,t)$ given by

$$u_t - u_{xx} + u = f(x,t) \quad 0 < x < 1 \tag{7}$$

with the boundary conditions $u(0) = 0$ and $u'(1) = 4$ and the initial condition $u(x,0) = g(x)$.

a. Write a weak formulation (not the discrete weak form) for this problem and show that a solution to the given initial value problem (7) is a solution of your weak formulation. Be sure to clearly define the space in which the solution is assumed to exist.

b. To discretize, assume that we subdivide $[0,1]$ into $n+1$ equal subintervals by the points $0 = x_0 < x_1 < ... < x_n < x_{n+1} = 1$, and that we plan to use piecewise linear basis functions $\{\phi_i(x)\}$ to approximate $u(x,t)$ by $u^h(x,t)$. Further assume that a backward Euler approximation is used in time.

(i) Show that at each time step we must solve a linear system of equations. Give an expression for each nonzero entry A_{ij} in the ith row of the coefficient matrix.

(ii) Explain how you satisfy the boundary conditions.

(iii) Give the size and bandwidth of the resulting matrix. Justify why your entries in the coefficient matrix outside of the bandwidth are zero.

(iv) If a uniform time step is used explain how would you efficiently solve the linear system at each time step.

c. The mass matrix M has entries $M_{ij} = \int_0^1 \phi_i(x)\phi_j(x)\,dx$. If the integrals are integrated exactly, determine the third row of the mass matrix for this discretization using piecewise linear basis functions.

10. **Stability and convergence of numerical PDEs** (Plewa)

Consider an explicit numerical solution of a one-dimensional advection problem; see

`http://people.sc.fsu.edu/~jburkardt/c_src/fd1d_advection_lax_wendroff/`
`fd1d_advection_lax_wendroff.html`

The above website provides implementations of the Lax method in C, C++., and Fortran 77/90, including the initial and boundary conditions for a smooth test problem.
You can either use one of the above codes or implement the Lax algorithm yourself using one of the above languages.
Compile the code and build an executable. Obtain, analyze, and discuss the results for the following problems.
Recall that for time-dependent problems, the solution error contains spatial, temporal, and mixed (spatio-temporal) terms. For example, using a suitable solution error norm, e.g. L_1, one can write

$$L_1 = A(dx)^\alpha + B(dt)^\beta + C(dxdt)^\gamma \, .$$

Your task is to estimate the coefficients A, B, C, and exponents (rates of convergence)α, β, and γ for the above Lax-Wendroff code and test problem implementation. To this end, you need to modify the code and perform a series of experiments methodically varying mesh resolution (in space and/or time). To determine coefficients and powers in the above model equation use a chi-square fit to data.
Make sure your spatial convergence experiments are performed when the code runs stably at all resolutions (i.e., the CFL conditions is always satisfied) and the solution is obtained in an asymptotic regime (loosely speaking the solution is well-resolved and the above error model applies).
For spatial experiments, you may want to start with a coarse mesh containing only 8 cells (nx=9), then double the mesh resolution until you have a few data points in the asymptotic regime allowing for extracting A and α. For temporal experiments, choose a well-resolved model and vary the time step. Your space-time study requires more creative thinking. (Hint: It has to be performed last.)
Document your experiments by showing a table with mesh resolution and corresponding error norms. Document your analysis by graphing data and model fit. Discuss the results.

(1) Spatial convergence. (50%).
(2) Temporal convergence. (30%).
(3) Spatio-temporal convergence (interaction term). (20%).

11. **Parallelization and performance of a molecular dynamics code** (Plewa)
Consider an OpenMP implementation of a molecular dynamics algorithm,

`http://people.sc.fsu.edu/~jburkardt/c_src/md_openmp/md_openmp.html`

The above website provides implementations in C, C++., and Fortran 77/90. Select one of the implementations and perform the following tasks.

a. Compile and execute the original implementation and record wall clock times using 1 2, 4, and 8 cores. Use the code's stdout to verify it produces correct results when executed in parallel. What information is reported every 40 steps?

b. Parallelize the code using MPI. Verify the MPI version produces correct results using 1, 2, 4, and 8 cores. Record wall-clock times for each experiment.

c. Based on the code timings, estimate the amount of serial code in OpenMP and MPI versions.

Document your work by providing the MPI version and relevant code standard outputs. Show wall-clock times for both code versions in a single plot. For comparison, add a line showing a perfect scaling relation. Discuss differences in performance between parallel implementations.