
Department of Scientific Computing
Written Preliminary Examination
Summer 2010

July 22, 2010

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 the percentage is explicitly state for each part.

If you use web sources, please list them clearly.

The exam is due back to Cecilia Farmer no later than 1:00 pm on Monday, July 26, 2010; no exceptions allowed.

If you have any questions related to this exam as you work on it, please send an e-mail to Dr. El-Azab (aelazab at fsu dot 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.

---

1. *Ordinary differential equations*

Consider the first order initial value problem (IVP)

$$y'(t) = f(t, y), \qquad y(0) = y_0\,.$$

We know that the Forward Euler method for this problem can be written as

$$Y^{n+1} = Y^n + \Delta t f(t^n, Y^n)$$

where $Y^n \approx y(t^n)$, $Y^0 = y_0$ and $\Delta t = t^{n+1} - t^n$.

a. One way to derive the first order Forward Euler method for this problem is to use Taylor Series. We can also use Taylor Series to derive higher order schemes. Derive a third order Taylor Series method for this IVP. Explicitly demonstrate that the method is third order accurate. Why is a third order Runge-Kutta or multistep method preferable to this higher order Taylor scheme? You may assume that $f(t, y)$ is as smooth as you need it to be.

b. Another way to derive the Forward Euler scheme is to integrate the differential equation from t^n to t^{n+1} and replace the integral of $f(t, y)$ by a quadrature rule. For example, Forward Euler can be obtained by using a left Riemann sum to approximate the integral; specifically,

$$y(t^{n+1}) - y(t^n) = \int_{t^n}^{t^{n+1}} y'(t)\, dt = \int_{t^n}^{t^{n+1}} f(t, y) \approx f(t^n, y(t^n))\Delta t.$$

Derive a scheme which uses Simpson's rule to approximate the integral of $f(t, y)$ and explain how you would implement it. Is this scheme explicit or implicit? What is the accuracy of the method? Is the scheme a Runge-Kutta method, a multistep method or neither? Give your reasoning for each response.

---

2. *Linear algebra*

In each case determine an appropriate algorithm for numerically solving the given linear algebra problem. Carefully explain why this algorithm would work for the given problem and why you chose it over others with regard to considerations such as accuracy, stability, work required, etc. If you recommend an iterative method, indicate what you would choose for the initial guess.

a. The solution of $A\vec{x} = \vec{b}$ where A is an $n \times n$ real, symmetric, indefinite matrix with $\kappa_2(A) \approx 10^8$. Here $\kappa(A)$ denotes the condition number of A.

b. The eigenvalue nearest one of the real, symmetric, positive definite matrix A.

c. The solution of $A\vec{x} = \vec{b}$ where A is an $n \times n$ real, symmetric matrix with at most p nonzero entries in each row where $p << n$ and $n = \mathcal{O}(10^6)$.

d. A basis for the null space of A where A is an $n \times n$ real, singular matrix.

---

3. *Gaussian quadrature*

Consider the calculation of radial integrals of the form

$$I = R^3 \int_0^1 f(\rho R) 4\pi\rho^2 d\rho$$

in 3D using Gaussian quadrature. Calculate the weights and nodes for N=5, where N is the total number of nodes. Verify your answer.

---

4. *Monte Carlo*

Write a program to simulate a 3D random walk of N steps $\{\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_N\}$, and unit step length, ($|\mathbf{r}_i - \mathbf{r}_{i-1}| = 1$). Plot the mean end-to-end squared distance (below) as a function of N for N between 10 and 100.

$$R_e^2(N) = \langle (\mathbf{r}_N - \mathbf{r}_0)^2 \rangle$$

The angular brackets indicate a mean over several realizations.

---

5. *Computational Fourier analysis*

(a) Consider the signal

$$f(t) = \sin(50\pi t) + \sin(160\pi t) + \sin(250\pi t) + \sin(480\pi t) + \sin(630\pi t)\,.$$

(i) Corrupt the signal $f(t)$ with a zero-mean random noise and graph the corrupted and original signals over an appropriate interval. Comment on the difference between the two signals. (20%)

(ii) Use FFT method (via MATLAB or any other software you prefer) to obtain (and plot) the power spectrum of the corrupted signal and identify from it the frequencies of the original signal. Use $N = 2^9 = 512$. Show details of your work and comment on your answer. (40%)

(b) Consider the differential equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = h(x, y),$$

where $h(x, y)$ is periodic in x and y with period L. Show (in steps) how discrete Fourier transform technique can be used to solve this differential equation. (40%)

---

6. *Partial differential equations: finite element method*

Consider the diffusion equation

$$f_t = \alpha \nabla^2 f(\mathbf{x}, t) + s(\mathbf{x}, t),$$

with $\mathbf{x} \in \Omega \subset \mathbb{R}^3$, $t \in [0, T]$, for some time interval T and positive value of the parameter α.

(a) State two physical situations where this equation applies and explain the significance of various terms in the two cases. (5%)

(b) If a Neumann boundary condition is to be applied over $\partial\Omega$ in the steady state, i.e., when $f_t = 0$ and in the absence of the source term s, what constraint must that boundary condition satisfy? Even with this constraint being satisfied, can one obtain a numerical solution in this case without imposing any additional condition? Explain. (20%)

Consider again the diffusion equation with non-trivial f_t and s. Simplify this equation to a 1D situation and consider the initial and boundary conditions

$$f(x, 0) = h(x), \quad f(0, t) = f_L, \quad f(1, t) = f_R.$$

The functions $s(x, t)$ and $h(x)$ are prescribed over the interval $0 < x < 1$, and f_L and f_R are constants.

(c) Derive a complete forward-time centered-space (FTCS) finite difference scheme to solve this initial-boundary value problem over the interval $0 < x < 1$. Use the notation Δx and Δt for grid size in space and time, respectively. Describe (in steps) how you will apply this scheme to solve the above problem for an arbitrary number of time steps. (40%)

(d) Formally analyze the consistency and order of the FTCS scheme of part (c). (35%)

---

7. *Optimization*

Consider the BFGS Quasi-Newton (Q-N) algorithm with Q-N update given by

$$\text{(BFGS)} \qquad B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}.$$

The algorithm starts with initial Hessian approximation $B_\circ = I$; I being the unit matrix. For $k = 0, 1, \ldots,$ if x_k is optimal, stop.

(i) Solve $B_k p_k = -\nabla f(x_k)$ for p_k, where p_k is the search direction. Use a line search to determine $x_{k+1} = x_k + \alpha_k p_k$, where α_k is the step length.

(ii) Compute intermediary quantities

$$s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

then compute

$$\text{(BFGS)} \qquad B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

(iii) Perform 2-3 iterations for BFGS Q-N for the 3-D quadratic problem:

$$f(x) = \frac{1}{2} x^T Q x - c^T x \quad \text{with} \quad Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix}$$

whose exact solution is $x^* = (-4, -3, -2)^T$. We will use for this quadratic problem an exact line search:

$$\alpha = -\frac{p^T \nabla f(x)}{p^T Q p}$$

with $B_\circ = I$ and $x_\circ = (0, 0, 0)^T$. At the first iteration of the Q-N BFGS, $\|\nabla f(x_\circ)\| = 14.4568$. Hence this point is not optimal. We must continue the iterative process. The first search direction can be obtained by taking the derivative of the quadratic function $f(x)$. Show that we finally obtain after 3 Q-N BFGS iterations:

$$x_3 = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix} \quad \text{and} \quad B_3 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

and now $\|\nabla f(x_3)\| = 0$, i.e., we have a local (in this case global) minimum. Why did Q-N BFGS converge in 3 iterations? Explain.

---

8. *Parallel programming*

Write an OPENMP program in C or Fortran to compute all the prime numbers between 1 and 1000000. You will get full credit if your code satisfies the following condtions:
1. It is parallel.
2. Your code is supposed to be faster when run on a machine with more CPUs.
3. It outputs all the prime numbers correctly.

---

9. *Numerical ODE*

The higher-order ODE below,

$$\frac{d^2h}{dx^2} = \frac{C}{T}(h - h_{wt}), \quad h_{wt} = 100 + 0.06x - 0.00003x^2,$$

is defined for a one-dimensional domain from $x = 0\,\text{m}$ to $x = 1000\,\text{m}$. The coefficients $T = 2.5 \times 10^{-5}$ m^2s^{-1} and $C = 10^{-9}$ s^{-1} are constant over the domain. The boundary condition at the left boundary is $h(x = 0) = 100\,\text{m}$.

Answer the questions below for different boundary conditions:

(i) Initial-value problem (25%)

Given the boundary condition at the left boundary, $q_w = -T\frac{dh}{dx}|_{x=0} = 10^{-7}\,\text{m}^2\text{s}^{-1}$, transform the initial-value problem of second-order ODE into a set of first-order ODEs. Solve the first-order ODEs numerically (e.g., using MATLAB) and plot h with x.

(ii) Boundary-value problem (25%)

Ignore the boundary condition of q_w above, and use the boundary condition $h(x = 1000\,\text{m}) = 93\,\text{m}$ at the right end of the domain. Solve the second-order ODE of this boundary condition using the shooting method. Plot h with x for the first and second guess and the final solution.

(iii) Finite difference (50%)

The boundary condition at the right end of the aquifer ($x = 1000\text{m}$) now becomes $\frac{dh}{dx}|_{x=1000} = 0$. Solve the second-order ODE using the finite difference method and plot h with x.

10. *Data structure*

I am interested in solving a 2-D differential equation on complex physical domain. To this end, I generated a multi-domain grid (i.e., the physical domain is covered by a collection of simpler curvilinear meshes such that each point in the physical domain is within one or more of these meshes). From a programmatic point of view, the grid is described by the following datastructure (I use a C++ structure, but could also have been Java or Fortran 90):

```
class Grid {
private:
  int nx, ny;   // number of points in the i and j directions
  float **x;    // x(i,j), i = 0,...,nx-1
  float **y;    // y(i,j), j = 0,...,ny-1
  float area;   // area covered by the grid
            // (precomputed once the grid is known)
  ...           // additional variables, not of relevance here
public:
  ...
};

int nb_grids;
Grid* grid;
```

The main program will take care of actually constructing `nb_grids` `Grid` structures. Of course, they will be in random order. Thus, the program maintains a list:

```
std::vector<Grid> gridList;
```

with the areas in random order. This vector will contains tens of thousand of grids (for a large-scale AMR method for example).
For reasons of efficiency, I am interested in sorting these grids in order of increasing area. Thus, I require a new list:

```
std::vector<Grid> sortedGridList;
```

where the grid areas are sorted.
The numerical algorithm requires finding the `Grid` structure whose area is closest to a specified area A. This operation will be required millions, if not more, times.

(a) (35pts) Write a method that generates `sortedGridList` starting from `gridList`. What is the asymptotic complexity of your algorithm? The chosen algorithm must be faster than $O(n^2)$. (hash tables not allowed)

(b) (35pts) Assuming that the adaptive grid is static (does not change during the course of the algorithm), write a method in C++ with signature

```
Grid* getGrid(float grid_area);
```

that returns the grid whose area is closest to `grid_area`, using `sortedGridList`, defined above. `grid_area` is an arbitrary area. The algorithm must be faster than $O(n)$. What is the asymptotic complexity of the algorithm? (hash tables not allowed.)

(c) (30pts) What types of datastructure problems would benefit from the use of hash tables? What are the advantages and disadvantages of hash tables with respect to linked lists?

There is no need to actually code up the answers to the questions, although you can if you have the time (no additional points will be given though).

---

11. *Numerical differentiation* Consider the function

$$f(x) = e^x$$

(a) (45pts) For an appropriate series of values for h estimate the derivative of at x using the first order formula

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

and estimate the value of h that minimizes the error in the estimate of the derivative (without using the computer). Be sure your series of h values is chosen to clearly support your conclusion of which h value minimizes the error. Compare this value of h against the value found from computing the error numerically and plotting the error as a function of $\log(h)$, where $h = 10^{-10}$ to $h = 10^{-1}$. Do all calculations and computations in single precision.

(b) (35pts) Repeat (a) in double precision.

(c) (20pts) for parts (a) and (b), explain why the errors in first decrease with decreasing h and then, at some value of h, begin to increase as h decreases further.
