Department of Scientific Computing

Written Preliminary Examination

Summer 2016

May 24 - 27, 2016

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 you use web sources, please list them clearly.
- The exam is due back to Mark Howard no later than 1:00 pm on Friday, May 27, 2016; 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 at fsu dot edu). The person responsible is listed at the beginning of each question.
- 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. **Numerical Linear Algebra** (Dr. Peterson)

In this problem we consider iterative methods for solving the invertible system $A\vec{x} = \vec{b}$.

(a) A stationary iterative method for solving the linear system is typically written as

$$\vec{x}^{k+1} = P\vec{x}^{k} + \vec{c} \tag{1}$$

whereas a non stationary iterative method is typically written as

$$\vec{x}^{k+1} = \vec{x}^{k} + \sigma_k \vec{d}_k \, .$$

Clearly describe the differences between stationary and non stationary iterative methods. Give three examples of stationary algorithms and three examples of non stationary; just give their names.

(b) Consider the splitting of the matrix A given by

$$A = T + L + U$$

where T is the tridiagonal portion of A, L is the remaining lower triangular portion of A and U is the remaining upper triangular portion of A; for example,

$$T = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 & 0 & 0 \\ 0 & 0 & A_{43} & A_{44} & A_{45} & 0 & 0 \\ & & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & 0 & A_{n,n-1} & A_{nn} \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ A_{31} & 0 & 0 & 0 & 0 & 0 \\ A_{41} & A_{4,2} & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & & & \\ A_{n1} & A_{n2} & \cdots & A_{n,n-2} & 0 & 0 \end{pmatrix}.$$

Use this splitting to write the iterative method in the form of equation (1); explicitly give the matrix P and the vector $\vec{c}$.

(c) Recall how the SOR algorithm was obtained from the Gauss-Seidel algorithm and use this concept in an analogous way to write a new algorithm where your algorithm in (b) plays the role of the Gauss-Seidel method.

(d) Give pseudo-code to describe how the method in (b) can be *efficiently* implemented. Indicate the storage used so that the minimum amount is allocated. Give the order of operations required to perform p iterations assuming that A is $n \times n$.

(e) Write a code to implement this method. Test your code on a range of full matrices which have various properties to make an educated guess for what type of matrices this algorithm converges. For example, start with a symmetric positive definite matrix and finish with a general matrix.

Point distribution: (a) - 15; (b) - 20; (c)- 10; (d) - 25; (e) - 30

Q2. **Approximation** (Dr. Shanbhag)

Approximate the function

$$f(x) = \frac{\sin 5x}{x},$$

on the interval $[-1, 1]$ using a monomial basis $\{1, x, x^2, \ldots, x^n\}$, by minimizing the L_2 norm:

$$\int_{-1}^{1} [f(x) - p_n(x)]^2 \, dx, \quad \text{where } p_n(x) = \sum_{i=0}^{n} a_i x^i.$$

1. For $n = 4$, write down the resulting linear system (**Hint**: think Hilbert matrix) $\mathbf{Aa} = \mathbf{f}$, where $\mathbf{a} = [a_0, a_1, ..., a_4]^T$, and solve it. Report $p_4(x)$. Use numerical quadrature (using built-in/public-domain functions), if and as required, to set up the problem. (40 points)

2. Plot $f(x)$ and compare it with $p_4(x)$. Plot $|f(x) - p_4(x)|$ as well. (20 points)

3. Suppose we use Legendre polynomials as a basis, so that:

$$p_n(x) = \sum_{i=0}^{n} c_i P_i(x),$$

where $P_i(x)$ are Legendre polynomials, for which the inner-product,

$$\langle P_i, P_j \rangle = \int_{-1}^{1} P_i(x) P_j(x) dx = \frac{2}{2i+1} \delta_{ij}.$$

Using the idea of inner-products, write an algorithm to project $f(x)$ onto the Legendre polynomial basis, and determine the coefficients c_i for $n = 4$. That is, use:

$$\begin{aligned} \langle f, P_j \rangle &= \sum_{i=0}^{n} c_i \langle P_i, P_j \rangle \\ &= c_i \langle P_i, P_i \rangle, \end{aligned}$$

where

$$\langle f, P_j \rangle = \int_{-1}^{1} f(x) P_j(x) dx.$$

(40 points)

Note: The first few Legendre polynomials are,

$$\{1, x, \frac{1}{2}(3x^2 - 1), \frac{1}{2}(5x^3 - 3x), \frac{1}{8}(35x^4 - 30x^2 + 3)\}$$

Q3. **ODEs and Linear Algebra** (Dr. Shanbhag)

Consider a homogeneous linear system, with $\mathbf{y} = [y_1, y_2]^T$,

$$\mathbf{y}' = \mathbf{A}\mathbf{y}; \quad \mathbf{y}(0) = [1, 4]^T,$$

on the domain $0 \leq t \leq 1$. If $\mathbf{A}$ is nondefective, then the analytical solution is given by:

$$\mathbf{y}(t) = c_1\mathbf{v}_1 e^{\lambda_1 t} + c_2\mathbf{v}_2 e^{\lambda_2 t},$$

where $\mathbf{v}_i$ and λ_i, ($i = 1$ and 2) are the eigenvectors and eigenvalues of $\mathbf{A}$.

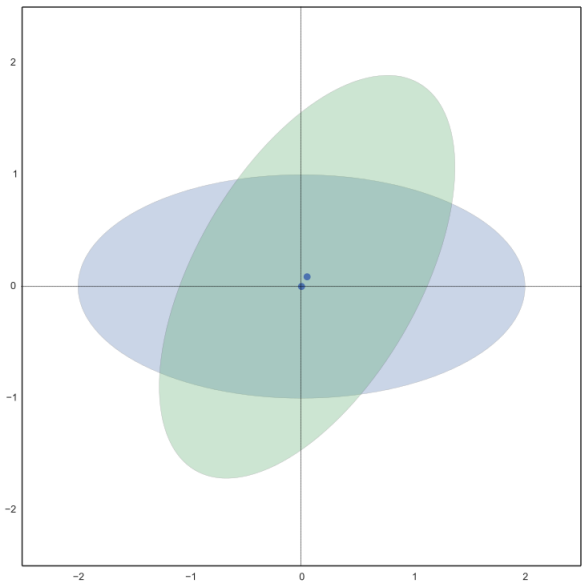
1. For
$$\mathbf{A} = \begin{bmatrix} -10.9 & -29.7 \\ -29.7 & -90.1 \end{bmatrix},$$
find c_1 and c_2 by using the initial conditions, and the analytical solution. (30 points)

2. Write down a second order accurate explicit Runge Kutta scheme, and use the test equation method to determine the maximum step size h^* for stability *for this problem.* (40 points)

3. Integrate the above system with a time-step slightly smaller and larger that it's theoretical maximum and comment on the behavior of your solution. (30 points)

Q4. **Monte Carlo** (Dr. Shanbhag)

Consider an ellipse E_1 with semi-major axis $a = 2$, and semi-minor axis $b = 1$, centered at the origin $(0, 0)$, and oriented with its major axis along the x-axis. Consider a second ellipse E_2, which has the same shape as E_1, but is oriented at an angle $\theta = \pi/3$ with respect to the x-axis. Furthermore, the center of E_2 is at a distance $\delta = 0.1$ away from the origin.



Find the area of overlap between E_1 and E_2 using Monte Carlo.

Q5. **ODE and Integral** (Dr. Quaife)

1. Construct the unique quadratic polynomial, $p(t)$, that interpolates a function f at the three points $(0, f(0))$, $(1/2, f(1/2))$, $(1, f(1))$. Compute the three integrals

$$\int_0^0 p(t)dt, \quad \int_0^{1/2} p(t)dt, \quad \int_0^1 p(t)dt.$$

   (20 points)

2. These integrals can be used to approximate the integral of f. Write down the matrix S for the linear system

$$S \begin{bmatrix} f(0) \\ f(\frac{1}{2}) \\ f(1) \end{bmatrix} = \begin{bmatrix} \int_0^0 f(t)dt \\ \int_0^{\frac{1}{2}} f(t)dt \\ \int_0^1 f(t)dt \end{bmatrix}$$

   that corresponds to this approximation. (20 points)

3. Consider the initial value problem $\dot{y}(t) = \lambda y(t)$, $y(0) = 1$. Given the solution at time t, the solution at some later time $t + \Delta t$ is

$$y(t + \Delta t) = y(t) + \int_t^{t+\Delta t} \lambda y(\tau)d\tau.$$

   Using the matrix S, we can approximate $y(t + \Delta t)$ by solving

$$\begin{bmatrix} y(t) \\ y(t + \frac{\Delta t}{2}) \\ y(t + \Delta t) \end{bmatrix} = \begin{bmatrix} y(t) \\ y(t) \\ y(t) \end{bmatrix} + \Delta t \lambda S \begin{bmatrix} y(t) \\ y(t + \frac{\Delta t}{2}) \\ y(t + \Delta t) \end{bmatrix}. \tag{2}$$

   Use any method, such as Cramer's rule, to solve equation (2) for $y(t + \Delta t)$ in terms of $y(t)$. (40 points)

4. Part 3 gives a time stepping method to move from time t to time $t + \Delta t$. Perform a convergence study with $\lambda = 2$ to show that fourth-order convergence is achieved. (20 points)

Q6. **Numerical Integration** (Dr. Burkardt)
Background: A numerical quadrature rule Q for a 2-dimensional region Ω is a set of n triples $(w_i, x_i, y_i), i = 1, ..., n$, where w_i is a *weight* and (usually) $(x_i, y_i) \in \Omega$. Suppose $f(x, y)$ is an integrable function defined over Ω, and symbolize its integral by $I(f) = \int_\Omega f(x, y)\, d\Omega$. The quadrature rule may be used to approximate such integrals by

$$I(f) \approx Q(f) \equiv \sum_{i=1}^{n} w_i f(x_i, y_i)$$

Let Ω be the *unit quarter disk*, that is, the set of points such that

$$\Omega = \{(x, y) | 0 <= x \text{ and } 0 <= y \text{ and } x^2 + y^2 <= 1\}$$

Using the notation $\Gamma(x)$ for the standard mathematical gamma function, it can be shown that, for nonnegative integer exponents p and q, we have:

$$I(x^p y^q) = \Gamma(\frac{p+3}{2})\Gamma(\frac{q+1}{2})/\Gamma(\frac{p+q+4}{2})/2/(1+p)$$

The total degree d of a monomial $x^p y^q$ is defined as $d = p + q$. We say that a quadrature rule for a 2-dimensional region has *precision* d if $I(f) = Q(f)$ for all monomials of total degree d or less.

A: (15 pts) We wish to compute a quadrature rule Q for Ω which has precision 2, using $n = 3$ points.

- A1: List the monomials which Q must integrate exactly.
- A2: What is the value of m, the number of monomials which Q must integrate exactly?
- A3: What is the value of k, the number of degrees of freedom we have in specifying the rule Q?

B: (30 pts) The rule Q must integrate the monomial of total degree 0, the two monomials of total degree 1, and the three monomials of total degree 2 exactly.

- B1: Write these statements down as a system of nonlinear equations involving the data $(w_i, x_i, y_i), i = 1, n)$. Where convenient, you may use symbolic expressions rather than numeric values.
- B2: Write a computer function **resid()** that accepts arbitrary values for the data $(w_i, x_i, y_i), i = 1, n)$ and returns the vector r of residuals, that is, $r(1) = Q(1) - I(1), r(2) = Q(x) - I(x), ...$ Include a copy of your **resid()** function with your answers.

C: (40 pts) Write a program that guesses initial values for Q, and then solves the equations $r = 0$ by calling some appropriate builtin-in library function for the solution of a rectangular $(m \times k)$ system of nonlinear equations. The MATLAB functions $fsolve()$ or $lsqnonlin()$ are examples of such software.

- C1: What routine did you use to solve the system of nonlinear equations?

- C2: What initial values did you use for the points and weights?
- C3: What final values did your program return for the points and weights?
- C4: Include a copy of your program with your answers.

D: (15 pts) Let $f(x) = 4 + 8xy$.

- D1: What is your estimated integral Q(f)?
- D2: Assuming exact arithmetic, should $Q(f) = I(f)$ in this case?

Q7. **PDE and Finite Difference** (Dr. Burkardt)
Background: A typical one-dimensional boundary value problem might be written as

$$-\frac{d}{dx}\left[a(x)\frac{du(x)}{dx}\right] + c(x)u(x) = f(x) \text{ for } 0 < x < 1$$
$$u(0) = 0; u(1) = 0$$

where $a(x), c(x), f(x)$ are given functions. In order to approximate such a problem with finite differences, it is necessary to rewrite the equation as

$$-a(x)\frac{d^2u(x)}{dx} - \frac{da(x)}{dx}\frac{du(x)}{dx} + c(x)u(x) = f(x)$$

Assuming we use n equally spaced nodes for our finite difference scheme, then at each interior node, we can replace $\frac{d^2u(x)}{dx^2}$ and $\frac{du(x)}{dx}$ by centered difference approximations, involving the left and right neighbor nodes, while evaluating $a(x), \frac{da(x)}{dx}, c(x), f(x)$ at the node. The solutions at the first and last nodes are zero, from the boundary conditions.

If we use n equally spaced nodes in $[0, 1]$, then the mesh size is $h = \frac{1}{n-1}$.

If u_i^h is the approximate solution at node x_i and u is the exact solution, then the l2 error will be

$$e(h) = \sqrt{\sum_{i=1}^{n}(u_i^h - u(x_i))^2}$$

Given solutions computed at h_1 and a smaller mesh size h_2, we say the observed l2 convergence rate is

$$\text{rate} = \log(e(h1)/e(h2))/\log(h1/h2)$$

A: (50 pts) Suppose we have a boundary value problem for which:

- $a(x) = 1 + x^2$;
- $c(x) = 2$;
- $f(x) = (3x + x^2 + 5x^3 + x^4)e^x$;
- $u(x) = (x - x^2)e^x$.

Write a program which is given n, the number of nodes, and computes the solution u^h and the l2 error $e(h)$. Include a copy of your program with your answers.

B: (25 pts) Using the value $n = 9$, compute the solution to the problem, and print a table of $i, u_i^h, u(x_i)$.

C: (25 pts) Compute a sequence of five approximate solutions, starting with $n = 9$ and reducing the mesh size by half each time.

- C1: Print a table of your mesh sizes h and l2 errors $e(h)$;
- C2: Compute and print 4 convergence rates by comparing successive pairs of l2 errors.

Q8. **Parallel Computing** (Dr. Huang)

The calculation of the product between a matrix B and a vector $\vec{a}$, i.e., $\vec{c} = B\vec{a}$, can be parallelized using MPI. The parallel efficiency is often very good. Suppose that the matrix B is of the size $n \times m$ and the vector $\vec{a}$ is of the size $m \times 1$.

1. If we have n processors, we can let the processor $j(1 \leq j \leq n)$ handle the dot product between the j^{th} row of matrix B and the vector $\vec{a}$, i.e., $\sum_{1 \leq q \leq m} B_{jq} a_q$. Write a MPI-based code to distribute the task of computing $B\vec{a}$ to the n processors by using the MPI function MPI_ALLREDUCE(). You are not required to run and debug your code, but try to make sure that your code is bug-free.

2. By using MPI_ALLREDUCE(), $\vec{c}$ is updated on all processors. If we only want to update $\vec{c}$ on the processor 0, we can let all the other processors send their results to the processor 0. Modify your code to accomplish this task by using MPI functions: MPI_SEND() and MPI_RECV().

3. If the number of processors, r is smaller than n, we can balance the computational cost among processors by letting each processor handle y rows of matrix B, where $y = \text{floor}(n/r)$. This can be a good choice, if n is divisible by r. Modify your code to realize this idea.

4. Discuss how to distribute the computational costs, if n is not divisible by r. Modify your code to realize your idea.

Q9. **Fast Fourier Transform** (Dr. Meyer-Baese)

For a common factor FFT the following 2D DFT is used:

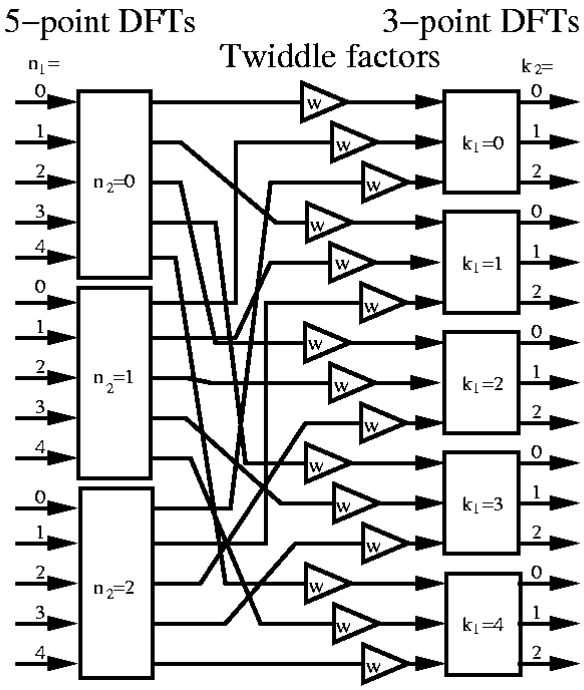
$$X[k_1, k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left(W_N^{n_2 k_1} \sum_{n_1=0}^{N_1-1} x[n_1, n_2] W_{N_1}^{n_1 k_1} \right)$$

a) Complete the following table for the index map for a $N = 15$ with $N_1 = 5$ and $N_2 = 3$ FFT with: $n = 3n_1 + n_2$ and $k = k_1 + 5k_2$.

n_2	n_1				
	0	1	2	3	4
0					
1					
2					

k_2	k_1				
	0	1	2	3	4
0					
1					
2					

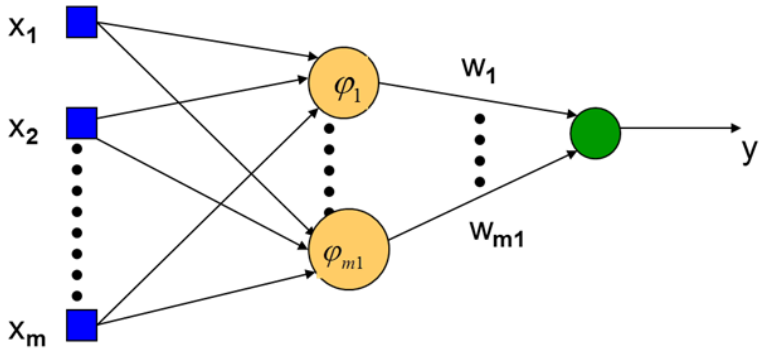
b) Complete the SFG (for x, X and and twiddle factors) for the FFT:



Q10. **Data Structure** (Dr. Meyer-Baese)
Assume we have a radial basis neural network (RBNN) as given below with the input layer having m neurons x_i, the hidden layer with $m1$ neurons (called also radial basis fields or receptive fields) having as an activation function $\phi_1 \cdots \phi_{m1}$ and weights w_i, one output neuron y:

$$y = w_1\phi_1(\|x - t_1\|) + \cdots + w_{m1}\phi_{m1}(\|x - t_{m1}\|)$$

The architecture of the RBNN is given in the following figure:



As activation functions ϕ_i we consider with r being the radius:
(a) An inverse multiquadrics with $c > 0$

$$\phi(r) = \frac{1}{\sqrt{r^2 + c^2}}$$

(b) A Gaussian function with the variance σ:

$$\phi(r) = \exp(-\frac{r^2}{2\sigma^2})$$

Please answer the following questions. Libraries cannot be used. You can use any language-like for the pseudocode (C++, Java, Python, Fortran, Matlab).

1. Assume that the activation function ϕ_i is both a Gaussian and an inverse multiquadrics function. Express the solution of the output layer as a function of x at the input layer as a matrix equation. Please explain all notation and reasoning. Use the given RBNN network above as an example. If the execution of ϕ_i counts as one operation, how many operations are required to find a solution? (30 pts)

2. Now consider a network with an input layer of size 10, with $m1$ hidden neurons or radial basis fields of Gaussian type. The output layer has p output nodes. How many operations are required to solve the network? Devise a notation and write down the matrix equation. Write a program to calculate the neural net. Assume that the weights are uniformly distributed between -1 and 1. Choose values of xi and σ_i to be random numbers between 0 and 1. Numerically calculate the computer time it takes to compute the outputs for $p = 5, 10, 15$. What is the scaling of time with respect to p? (35 pts)

3. Assume that the activation function of the radial basis field is a sigmoid function. Can we also give a matrix formulation of the RBNN? In the event that the matrix formulation is not appropriate, provide an alternative approach to computing the network (i.e., computing y)? Write some pseudocode. (20 pts)

4. Now, assume that we have a RBNN with a very large number of radial basis fields and with varying variance and center over time. Since the radial basis fields can change position over time, describe a data structure appropriate to the task. Please provide pseudocode that describes that data structure using object-orientated coding. (15 pts)

Q11. **Optimization** (Dr. Navon)
Here we are going to compare performance of Quasi-Newton DFP and BFGS methods.
Use the DFP quasi-Newton update of the inverse Hessian approximation:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{p_k p_k^T}{y_k^T p_k} \tag{3}$$

Starting with

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

being the unit matrix and minimize the function:

$$f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

with initial guess $X_1 = (-2, -2)^T$. Use accuracy condition

$$\|\nabla f_k\| < \epsilon \tag{4}$$

with $\epsilon = 10^{-3}$.
Compute the search direction $d_k = -H_k \nabla f_k$ and set $x_{k+1} = x_k + \alpha_k d_k$ where α_k is computed from a line search to satisfy Wolfe condition.
Define $p_k = x_{k+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$. Compute H_{k+1} by means of (3).
Check if accuracy condition (4) is satisfied, if not set $k \leftarrow k + 1$ and repeat the loop.
a) Please write your own code or use the code `http://people.sc.fsu.edu/~wwang3/dfpmintest-1.f` based on Numerical Recipes. Use ITMAX, maximum number of iterations as ITMAX = 200. If you use above code please flowchart it. Explain why the problem of minimizing Rosenbrock function is more difficult by considering its condition number.

Consider then BFGS quasi Newton method for same problem:

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

$$s_k = x_{k+1} - x_k$$

$$y_k = \nabla f_{k+1} - \nabla f_k$$

(BFGS) for the inverse Hessian approximation.
For the direct Hessian approximation we have via Sherman-Morrison-Woodbury formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

b) Use the code `http://people.sc.fsu.edu/~wwang3/conminf.f` or the Matlab code `http://people.sc.fsu.edu/~wwang3/ucminf.m` or write equivalent code by yourself and test it for the Rosenbrock function above, with same accuracy and starting point as that used for DFP quasi-Newton.
c) Use accuracy ϵ and compare results for same starting point

d) Plot in semi log both the function and gradient norm vs. the number of iterations for both DFP and BFGS.
e) Flowchart code of conminf.f or that of MATLAB code ucminf.m and explain its line-search method.
f) Please compare performance of DFP and BFGS in terms of number of iterations, accuracy achieved and norms of the gradient at convergence.
