CS412 Spring Semester 2012

Practice Final - Solutions

1. MULTIPLE CHOICE SECTION. Circle or underline the correct answer (or answers). You do not need to provide a justification for your answer(s).

   (1) Which of the following statements, regarding methods for solving a nonlinear equation $f(x) = 0$, are true?
   (Circle or underline ALL correct answers)

   (a) Newton's method is second order accurate, but requires knowledge of the derivative $f'(x)$.

   (b) The Secant method is second order accurate, and doesn't require knowledge of the derivative $f'(x)$.

   (c) The Bisection method offers superlinear convergence as long as $f'(\alpha) \neq 0$ at the exact solution $x = \alpha$.

   (2) Which of the following statements, regarding methods for solving a nonlinear equation $f(x) = 0$, are true?
   (Circle or underline ALL correct answers)

   (a) The Bisection method has linear convergence, and will safely converge to the solution, starting from an interval $[a, b]$, as long as $f(a)f(b) < 0$ and f is continuous.

   (b) Newton's method offers quadratic convergence, but is not guaranteed to always converge, especially if started from a value far from the solution.

   (c) The Secant method has superlinear (although not quadratic) convergence, but is always guaranteed to converge to a solution.

   (3) Which of the following are good reasons for using Lagrange interpolation?
   (Circle or underline ALL correct answers)

   (a) We can easily compute derivatives of the generated interpolant.

   (b) The interpolant can be easily defined, without solving large systems or performing expensive computations.

   (c) When applied to Runge's function, this method produces an interpolant that converges to the actual function with increasing number of interpolation points.

(4) Which of the following methods are *not* prone to oscillations, when interpolating through a large number of data points?
(Circle or underline ALL correct answers)

(a) Newton Interpolation.

(b) Polynomial interpolation using Chebyshev points.

(c) Low-order piecewise polynomials.

(5) Which of the following statements, regarding cubic splines, are true?
(Circle or underline ALL correct answers)

(a) The cubic spline method requires only function values, not derivatives, and produces a result with continuous second derivatives.

(b) Hermite splines require both function values and derivative values, but the computation is very easy and can be performed independently on each interval.

(c) Although Hermite splines are easier to compute, the interpolation error only scales proportionately to h^3, while cubic splines have the error decreasing proportionately to h^4.

(6) Which of the following statements, regarding the cost of methods for solving an $n \times n$ linear system $\mathbf{Ax} = \mathbf{b}$, are true?
(Circle or underline ALL correct answers)

(a) The cost of computing the LU factorization is generally proportional to n^2.

(b) The cost of backward substitution on a dense upper triangular matrix is generally proportional to n^2.

(c) If a matrix $\mathbf{A}$ has no more than 3 nonzero entries per row, the cost of each iteration of the Jacobi method is proportional to n.

(7) Which of the following statements, regarding numerical integration methods, are true?
(Circle or underline ALL correct answers)

(a) If the local error of an integration rule scales like $O(h^d)$, the global error will be $O(h^{d+1})$.

(b) With a second order accurate rule, if we increase the number of points in the integration rule by 10, we should expect the error to decrease by approximately a factor of 20.

(c) If a method computes the integral of polynomials up to order d exactly, then the global error is in the order of $O(h^{d+1})$.

(8) Which of the following statements, regarding methods for solving initial value problems, are true?
(Circle or underline ALL correct answers)

(a) Every step of an explicit method is very inexpensive, but these methods are restrictive in the maximum time step Δt that may be used.

(b) If a differential equation has unstable solutions, using an implicit method will guarantee convergence to the correct solutions, where explicit methods would diverge away from the real solution.

(c) Implicit methods generally require solving an equation for y_{k+1} at every step, while explicit methods typically provide the value of y_{k+1} directly as a function of known quantities.

2. CRITICAL THINKING PROBLEMS. Answer each of the following questions in **approximately 1 short paragraph**.

(a) Imagine that you want to generate a function $f(x)$ that interpolates a large number (hundreds or thousands) of data points (x_i, y_i). You also require to be able to compute derivatives of the interpolant, at arbitrary locations. Explain what technique you would employ, and why.

Answer Since we anticipate a very large number of data points, it would be preferable to use a piecewise polynomial method, since passing a single polynomial through all these data points would likely be expensive to compute and very sensitive to small changes in the location of the data points. Among piecewise polynomial interpolation methods, we would rather use cubic splines rather than Hermite splines, since we don't have derivative values at our disposal.

(b) Write the formula describing how the Trapezoidal Rule is used to solve the initial value problem $y' = -y + \sin(y)$. This should result in a nonlinear equation for y_{k+1}; write the formula for Newton's method, applied to solve this nonlinear equation.

Answer

$$y_{k+1} = y_k + \frac{\Delta t}{2}\left(-y_k + \sin(y_k) - y_{k+1} + \sin(y_{k+1})\right) \Rightarrow$$

$$\Rightarrow \left(1 + \frac{\Delta t}{2}\right) y_{k+1} - \frac{\Delta t}{2}\sin(y_{k+1}) - \left(1 - \frac{\Delta t}{2}\right) y_k - \frac{\Delta t}{2}\sin(y_k) = 0$$

This is a nonlinear equation of the form $f(y_{k+1}) = 0$, where

$$f(y) = \left(1 + \frac{\Delta t}{2}\right) y - \frac{\Delta t}{2}\sin(y) - \left(1 - \frac{\Delta t}{2}\right) y_k - \frac{\Delta t}{2}\sin(y_k)$$

$$f'(y) = 1 + \frac{\Delta t}{2} - \frac{\Delta t}{2}\cos(y)$$

Using the last 2 expressions, Newton's method yields:

$$y_{k+1}^{(n+1)} = y_{k+1}^{(n)} - \frac{f(y_{k+1}^{(n)})}{f'(y_{k+1}^{(n)})}$$

(c) Describe the reason why the QR factorization is an appealing method for solving least squares problems. Can you also describe a scenario where the method of normal equations is also a good alternative, even possibly preferable to the QR approach?

Answer The most important advantage of the QR factorization is that it may lead to a much better conditioned system than the normal equations. The QR method finds the least squares solution by solving the upper-triangular system $\mathbf{Rx} = \mathbf{Q}^T\mathbf{b}$; in this case the condition number of $\mathbf{R}$ is the square root of that of the matrix $\mathbf{A}^T\mathbf{A}$, which appears in the normal equations system.

However, when the condition number of $\mathbf{A}^T\mathbf{A}$ is not anticipated to be prohibitive, the system of normal equations can be a viable alternative, especially since it is very straightforward to compute, and the matrix $\mathbf{A}^T\mathbf{A}$ is symmetric and positive definite, which allows us to use not only direct, but also iterative methods to approximate its solution.

3. We have discussed that both the Backward Euler method, as well as the Trapezoidal rule are both *stable* methods for solving initial value problems of the form $y' = f(t, y),\ y(t_0) = y_0$. This excercise investigates some of the differences of these two stable methods.

   Consider the model stable equation $y' = \lambda y,\ \lambda < 0$. Write the equations defining the iteration of both the Backward Euler, and Trapezoidal rule methods. Consider that we decide to use a *very large* time step Δt. Show that when Δt is very large, the iterates produced by Backward Euler very quickly decay to zero, while the Trapezoidal rule produces oscillatory results, which ultimately also decay to zero.

   What are the consequences of this observation? Would this be a reason for prefering one method over the other in certain cases?

   Solution Applying Backward Euler to the model equation $y' = \lambda y,\ \lambda < 0$ gives:

$$y_{k+1} = y_k + \lambda \Delta t y_{k+1} \Rightarrow (1 - \lambda \Delta t) y_{k+1} = y_k \Rightarrow y_{k+1} = \frac{1}{1 - \lambda \Delta t} y_k$$

   Since

$$\lim_{\Delta t \to \infty} \frac{1}{1 - \lambda \Delta t} y_k = 0$$

   we conclude that when using the Backward Euler method with a very large Δt, the produced iterates $y_1, y_2, \ldots$ will almost immediately decay to zero (which, incidentally, is somewhat consistent with the exact solution $y(t) = e^{-|\lambda| t}$). In contrast, Trapezoidal rule yields:

$$y_{k+1} = y_k + \frac{\Delta t}{2}(\lambda y_k + \lambda y_{k+1}) \Rightarrow \left(1 - \frac{\lambda \Delta t}{2}\right) y_{k+1} = \left(1 + \frac{\lambda \Delta t}{2}\right) y_k \Rightarrow$$

$$\Rightarrow y_{k+1} = \frac{1 + \lambda \Delta t/2}{1 - \lambda \Delta t/2} y_k$$

   Now, in this case:

$$\lim_{\Delta t \to \infty} \frac{1 + \lambda \Delta t/2}{1 - \lambda \Delta t/2} y_k = \lim_{\Delta t \to \infty} \frac{1/\Delta t + \lambda/2}{1/\Delta t - \lambda/2} y_k = (-1) y_k = -y_k$$

   So, when a very large Δt is used, Trapezoidal rule has the effect that every iterate is approximately the negation $y_{k+1} = -y_k$ of the previous value.

   Since both Backward Euler and Trapezoidal Rule are stable methods, they will both ultimately produce a sequence y_k that converges to zero, as does the analytic solution. The difference is that, when using a large Δt, Backward Euler will converge to zero very rapidly, while Trapezoidal rule will oscillate between positive/negative values before settling down to zero.

4. The general form of an iterative method for solving the system $\mathbf{A}x = \mathbf{b}$ has the form

$$\mathbf{x}^{(k)} = \mathbf{T}\mathbf{x}^{(k-1)} + \mathbf{c}$$

where the matrix $\mathbf{T}$ and the vector $\mathbf{c}$ are such that the equation $\mathbf{x} = \mathbf{T}\mathbf{x}+\mathbf{c}$ is equivalent to the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

(a) If $\mathbf{x}$ is the exact solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, show that

$$\mathbf{x}^{(k)} - \mathbf{x} = \mathbf{T}\left(\mathbf{x}^{(k-1)} - \mathbf{x}\right)$$

(b) If $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ is the residual after the k-th iteration of the method, show that

$$\mathbf{r}^{(k)} = \mathbf{A}\mathbf{T}\mathbf{A}^{-1}\mathbf{r}^{(k-1)}$$

[Hint: Use the identity $\mathbf{r}^k = -\mathbf{A}\mathbf{e}^{(k)}$, or equivalently $\mathbf{e}^k = -\mathbf{A}^{-1}\mathbf{r}^{(k)}$ which we have proved in class. Here $\mathbf{e}^k = \mathbf{x}^{(k)} - \mathbf{x}$ is the *error* vector after the k-th iteration.]

(c) Show that

$$\mathbf{r}^{(k)} = \mathbf{A}\mathbf{T}^k\mathbf{A}^{-1}\mathbf{r}^{(0)}$$

Solution

(a) The exact solution satisfies $\mathbf{x} = \mathbf{T}\mathbf{x} + \mathbf{c}$. Subtracting this equation from $\mathbf{x}^{(k)} = \mathbf{T}\mathbf{x}^{(k-1)} + \mathbf{c}$ (which is the definition of the iterative method) we obtain

$$\mathbf{x}^{(k)} - \mathbf{x} = \mathbf{T}\left(\mathbf{x}^{(k-1)} - \mathbf{x}\right)$$

(b) The previous equation is equivalently written $\mathbf{e}^{(k)} = \mathbf{T}\mathbf{e}^{(k-1)}$. Since $\mathbf{r}^k = -\mathbf{A}\mathbf{e}^{(k)}$, or equivalently $\mathbf{e}^k = -\mathbf{A}^{-1}\mathbf{r}^{(k)}$, we can replace the error vectors in this equation with:

$$-\mathbf{A}^{-1}\mathbf{r}^{(k)} = -\mathbf{T}\mathbf{A}^{-1}\mathbf{r}^{(k-1)} \Rightarrow \mathbf{r}^{(k)} = \mathbf{A}\mathbf{T}\mathbf{A}^{-1}\mathbf{r}^{(k-1)}$$

(c)

$$\mathbf{r}^{(k)} = \mathbf{A}\mathbf{T}\mathbf{A}^{-1}\mathbf{r}^{(k-1)} = (\mathbf{A}\mathbf{T}\mathbf{A}^{-1})(\mathbf{A}\mathbf{T}\mathbf{A}^{-1})\mathbf{r}^{(k-2)} = \mathbf{A}\mathbf{T}^2\mathbf{A}^{-1}\mathbf{r}^{(k-2)}$$

$$= (\mathbf{A}\mathbf{T}^2\mathbf{A}^{-1})(\mathbf{A}\mathbf{T}\mathbf{A}^{-1})\mathbf{r}^{(k-3)} = \mathbf{A}\mathbf{T}^3\mathbf{A}^{-1}\mathbf{r}^{(k-3)} = \cdots = \mathbf{A}\mathbf{T}^k\mathbf{A}^{-1}\mathbf{r}^{(0)}$$

5. A convenient feature of the normal equations is that we can easily change them to account for additional equations that are being "added" to a least squares problem. For example, consider the following m equations (using n unknowns, with $m > n$):

$$\begin{gathered} c_1 x_1^{(1)} + c_2 x_2^{(1)} + \cdots + c_n x_n^{(1)} \approx y^{(1)} \\ c_1 x_1^{(2)} + c_2 x_2^{(2)} + \cdots + c_n x_n^{(2)} \approx y^{(2)} \\ c_1 x_1^{(3)} + c_2 x_2^{(3)} + \cdots + c_n x_n^{(3)} \approx y^{(3)} \\ \vdots \\ c_1 x_1^{(m)} + c_2 x_2^{(m)} + \cdots + c_n x_n^{(m)} \approx y^{(m)} \end{gathered}$$

We can write these equations as the least squares problem $\mathbf{Ax} \approx \mathbf{b}$ where

$$\mathbf{A} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & \cdots & x_n^{(3)} \\ \vdots & \vdots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}, \quad \mathbf{x} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

The least squares solution can be obtained by solving the normal equations

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$$

Now, assume that we want to add the following additional equation to our least squares problem

$$c_1 x_1^{(m+1)} + c_2 x_2^{(m+1)} + \cdots + c_n x_n^{(m+1)} \approx y^{(m+1)}$$

Show that the normal equations corresponding to this enlarged least squares problem take the form

$$(\mathbf{A}^T \mathbf{A} + \mathbf{w}\mathbf{w}^T)\mathbf{x} = \mathbf{A}^T \mathbf{b} + y^{(m+1)} \mathbf{w}$$

where $\mathbf{w} = [x_1^{(m+1)} \; x_2^{(m+1)} \; \cdots \; x_n^{(m+1)}]^T$.

[Hint: Write this new least squares problem, with $m+1$ equations, as $\hat{\mathbf{A}}\mathbf{x} \approx \hat{\mathbf{b}}$, and observe that $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{w}^T \end{bmatrix}$ and $\hat{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ y^{(m+1)} \end{pmatrix}$.]

Solution The *new* least squares problem, with the additional equation added, is written in matrix form as $\hat{\mathbf{A}}\mathbf{x} \approx \hat{\mathbf{b}}$, where:

$$\hat{\mathbf{A}} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & \cdots & x_n^{(3)} \\ \vdots & \vdots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \\ x_1^{(m+1)} & x_2^{(m+1)} & \cdots & x_n^{(m+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{w}^T \end{bmatrix}, \quad \hat{\mathbf{b}} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \\ y^{(m+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ y^{(m+1)} \end{pmatrix}$$

where $\mathbf{w} = [x_1^{(m+1)} \; x_2^{(m+1)} \; \cdots \; x_n^{(m+1)}]^T$.

The normal equations formed from this new system $\hat{\mathbf{A}}\mathbf{x} \approx \hat{\mathbf{b}}$ will be:

$$\hat{\mathbf{A}}^T\hat{\mathbf{A}}\mathbf{x} \approx \hat{\mathbf{A}}^T\hat{\mathbf{b}} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} \mathbf{A}^T & \mathbf{w} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{w}^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{A}^T & \mathbf{w} \end{bmatrix} \begin{pmatrix} \mathbf{b} \\ y^{(m+1)} \end{pmatrix} \Rightarrow$$

$$\Rightarrow (\mathbf{A}^T\mathbf{A} + \mathbf{w}\mathbf{w}^T)\mathbf{x} = \mathbf{A}^T\mathbf{b} + y^{(m+1)}\mathbf{w}$$