# Math 541, Study Guide

***— a set of topic you should be aware of, and questions which you should be able to answer at the end of the course.***

`$Id: studyguide.tex,v 1.9 2011/12/05 22:02:40 blomgren Exp $`

All these questions have been actual midterm/final questions at some point in history.

1. **Numerical Approximation:** Let $\tilde{x}$ be an approximation to $x$ ($x \neq 0$).
   (a) What is the absolute error?
   (b) What is the relative error?

2. **Fixed Point Iteration:** Consider the function $g(x) = 1 - 2x^2$, the fixed point equation $x = g(x)$, and the fixed point iteration $p_{n+1} = g(p_n), \quad p_0 \in [-1, 1]$.
   (a) Does the fixed point equation have any solution(s)? Is so, what are they?
   (b) If we start the fixed point iteration at $p_0 = 0.1$, is it likely it will converge to any of these solution(s)? Why? / Why not?

3. **Taylor Expansion, Newton's Method:** Consider the function $f(x) = e^x - \cos x - x$ on the interval $[0, 1]$.
   (a) Write the Taylor polynomial of degree 2, $P_2(x)$, and the remainder term, $R_2(x)$. [**Hint:** You can Taylor expand around any point, but $x_0 = 0$ makes life easier. $\frac{d}{dx}e^x = e^x$, $\frac{d}{dx}\cos(x) = -\sin(x)$, $\frac{d}{dx}\sin(x) = \cos(x)$ ]
   (b) Use the remainder term to get an upper bound on the error in the above approximation (in the given interval).
   (c) Now consider $g(x) = (f(x))^2 = (e^x - \cos x - x)^2$. What is the rate of convergence for $\lim_{h \searrow 0} g(h)$? [**Hint:** If you did part (a), this problem is easier than you fear!]
   (d) Write down Newton's method for $f(x) = 0$ (for the function $f$):
   (e) For *this particular function* $f$, what do you expect the rate of convergence to be (circle one): Slower than quadratic / Quadratic / Better than quadratic?
   (f) Why?
   (g) Write down a scheme which converges faster than the one you wrote down in (d). What is the convergence rate of this scheme?

4. **Newton's Method:** Numerical solution of $f(x) = 0$.
   (a) Write down Newton's Method.
   (b) Derive Newton's method using either a (i) geometric argument; (ii) Taylor series argument; or (iii) fixed-point argument. Clearly explain any approximations, and/or theorems that you use. ***Choose exactly one (1) approach!!!***
   (c) Assume $f(x^*) = 0$, and we have an initial approximation $x_0$ which is "close enough" to $x^*$. What is the convergence rate for Newton's method:
      i. If $f'(x^*) \neq 0$?
      ii. If $f'(x^*) = 0$?
   (d) How can we reclaim the quadratic convergence rate in the case where Newton's method for $f(x) = 0$ slows down? Explain carefully.
   (e) Assume $f \in C^2[a, b]$, and $x^* \in (a, b)$, so that $f(x^*) = 0$. Now view the Newton iteration as a fixed point iteration $x_{n+1} = g(x_n)$.

   i. What condition on $g(x)$ is necessary for convergence of the fixed point iteration?
   ii. What condition on $g'(x)$ is necessary for convergence of the fixed point iteration?
   iii. Translate the condition on $g'(x)$ to a condition for the Newton iteration. (Note: this is the expression which quantifies the neighborhood where $x_0$ is "close enough" to $x^*$.)

5. **Root Finding:** For each of the following methods, write down (i) the definition, (ii) What kind of point(s) we need to start the scheme? (iii) How fast does the scheme converge?
   (a) The Bisection Method
      i. Definition:
      ii. Starting Points:
      iii. Speed (in absolute terms):
   (b) The Secant Method
      i. Definition:
      ii. Starting Points:
      iii. Speed (compared with the other schemes):
   (c) Newton's Method
      i. Definition:
      ii. Starting Points:
      iii. Speed (in absolute terms):
   (d) Regula Falsi
      i. Definition:
      ii. Starting Points:
      iii. Speed (compared with the other schemes):
      iv. Impact of stopping criteria, for convex/concave functions.
6. **Multiplicity of zeros:** Know what it means, including impact on the derivatives, and how it effects (some) methods.
7. **Aitken's $\Delta^2$-method:** Given an already computed linearly convergent sequence $\{p_n\}_{n=1}^{\infty}$, how can we manufacture a sequence which converges faster?
8. **Steffensen's Method:** In general fixed-point iteration gives linear convergence (if it converges, that is) — recall the conditions for convergence of fixed point; when does it converge faster? With the help of ***Aitken's $\Delta^2$-method:*** we can generate a quadratically converging method which does not require computation of the derivative — how? Both Newton's and Steffensen's methods are quadratically convergent, so it seems like Steffensen would be our prime choice; why is it not? (That is, what additional restrictions (beyond what is needed for Newton) is needed for Steffensen's method to converge?
9. **Polynomials, Horner's Method:** Given the polynomial $P(x) = x^4 - x^3 + x^2 + x - 1$, use Horner's method (synthetic division) to *compute* $P(5)$ and $P'(5)$.
10. **Deflation, with Improvement:** The method for extracting ***all*** real roots of a polynomial.
11. **Müller's Method:** Understand it as a natural extension of the secant method, and how it automatically tells us when we get complex roots.
12. **The Lagrange Polynomial:** Understand how the building blocks $L_{n,k}(x)$ work, and how they allow us to interpolate any given data set $\{x_i, y_i = f(x_i)\}_{i=0}^{n}$.

13. **Newton's Divided Differences:** Given the points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \ldots, x_n\}$ and the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \ldots, f_n\}$, where $f_i = f(x_i),\ i = 0, 1, 2, \ldots, n$.

    (a) Explain how to fill in the table of Newton's Divided Differences — In particular, write down the expression for $F_{3,2}$ assuming all entries to the left in the table are known?

| $\vec{\mathbf{x}}$ | $\vec{\mathbf{f}}$ | 1st | 2nd. | $\cdots$ | $\cdots$ | $n$th. |
|---|---|---|---|---|---|---|
| $x_0$ | $f_0$ | | | | | |
| $x_1$ | $f_1$ | $F_{1,1}$ | | | | |
| $x_2$ | $f_2$ | $F_{2,1}$ | $F_{2,2}$ | | | |
| $x_3$ | $f_3$ | $F_{3,1}$ | $F_{3,2}$ | $F_{3,3}$ | | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | | $\ddots$ | |
| $x_n$ | $f_n$ | $F_{n,1}$ | $F_{n,2}$ | $\cdots$ | $\cdots$ | $F_{n,n}$ |

    (b) Once the table is full — how do we use it? Write down the interpolating polynomial of degree $n$ using the appropriate entries from the table.

14. **Hermite Interpolation:** Given the points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \ldots, x_n\}$, the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \ldots, f_n\}$, where $f_i = f(x_i),\ i = 0, 1, 2, \ldots, n$, and the values of the derivative $\vec{\mathbf{d}} = \{d_0, d_1, d_2, \ldots, d_n\}$, where $d_i = f'(x_i),\ i = 0, 1, 2, \ldots, n$.

    (a) Explain how to ***modify*** the table of Newton's Divided Differences to compute the Hermite interpolating polynomial:

| $\vec{\mathbf{x}}$ | $\vec{\mathbf{f}}$ | 1st | 2nd. | $\cdots$ | $n$th. |
|---|---|---|---|---|---|
| $x_0$ | $f_0$ | | | | |
| $x_1$ | $f_1$ | $F_{1,1}$ | | | |
| $x_2$ | $f_2$ | $F_{2,1}$ | $F_{2,2}$ | | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | |
| $x_n$ | $f_n$ | $F_{n,1}$ | $F_{n,2}$ | $\cdots$ | $F_{n,n}$ |

    i. First, explain how the table is initialized, *i.e.* how we use the values $\vec{\mathbf{x}}$, $\vec{\mathbf{f}}$, and $\vec{\mathbf{d}}$ to get started.

    ii. Second, how do we compute the rest of the values in the table? In particular, write down the expression for $F_{5,3}$ assuming all entries to the left in the table are known?

    iii. Once the table is full — how do we use it? Write down the interpolating polynomial of degree $(2n + 1)$ using the appropriate entries from the table.

---

**MIDTERM #1 MATERIAL ENDS HERE**

---

15. **Cubic Spline Interpolation:** Given the points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \dots, x_n\}$ and the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \dots, f_n\}$ at those points, we want to generate a **cubic spline**, *i.e.* a piecewise third degree polynomial approximation.

   (a) Why would we want to do this — why not just use Newton's Interpolatory Divided Difference formula to get an $n$th degree interpolating polynomial?.

   (b) Write down the interpolant $S_k(x)$ on the subinterval $[x_k, x_{k+1}]$.

   (c) Write down the conditions required for the spline $S(x)$ to fit the data and have two continuous derivatives, *i.e.* $S(x) \in C^2[x_0, x_n]$.

   (d) How many unknown coefficients do we have to determine?

   (e) How many equations do we have (in (c))?

   (f) Suggest additional boundary conditions, giving enough additional equations to close the system (assume we have no additional information about $f$).

16. **Piecewise Polynomial Approximation, Quintic Splines:** Given the points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \dots, x_n\}$ and the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \dots, f_n\}$ at those points, we want to generate a **quintic spline**, *i.e.* a piecewise fifth degree polynomial approximation.

   (a) Why would we want to do this — why not just use Newton's Interpolatory Divided Difference formula to get an $n$th degree interpolating polynomial?.

   (b) Write down the interpolant $S_k(x)$ on the subinterval $[x_k, x_{k+1}]$.

   (c) Write down the conditions required for the spline $S(x)$ to fit the data and have four continuous derivatives, *i.e.* $S(x) \in C^4[x_0, x_n]$.

   (d) How many unknown coefficients do we have to determine?

   (e) How many equations do we have (in (c))?

   (f) Suggest additional boundary conditions, giving enough additional equations to close the system (assume we have no additional information about $f$).

17. **Numerical Integration:** Simpson's Rule and Composite Simpson's Rule.

   (a) We write Simpson's Rule:

   $$\int_a^b f(x)\,dx \approx S(f(x), a, b) + \frac{h^5}{90} f^{(4)}(\xi) = \frac{h}{3}\Big[f(a) + 4f((a+b)/2) + f(b)\Big] + \frac{h^5}{90} f^{(4)}(\xi).$$

   Explain how we arrive at this formula using the *Lagrange Interpolating Polynomial* of degree 2. **IGNORE THE ERROR TERM, AND DO NOT COMPUTE ANY INTEGRAL(s)**.

   (b) We define the Composite Simpson's Rule by splitting the interval $[a, b]$ into smaller sub-intervals, applying Simpson's Rule on those sub-intervals, and then summing up the results. Write down Composite Simpson's Rule applied to the sub-intervals $[x_0, x_2]$, $[x_2, x_4]$, $[x_4, x_6]$, $[x_6, x_8]$, where $x_k = a + \frac{k(b-a)}{8}$.

   (c) **Richardson Extrapolation:** Explain how we can use multiple instances of Composite Simpson's Rule (with point-spacing $h$, $h/2$, $h/4$, ...) to generate a scheme with an error term $\sim \mathcal{O}(h^6)$. (Warning: this question is harder than it looks!)

18. **Numerical Integration, Gaussian Quadrature:** We are interested in integrating a function of two variables $f(x, y)$ over the square $[-1, 1] \times [-1, 1]$, *i.e.* $\mathcal{I} = \int_{-1}^{1}\int_{-1}^{1} f(x, y)\,dx\,dy$. Build a 9-point numerical integration scheme based on 3-point Gaussian quadrature in the $x$- and $y$-directions. Specify the quadrature points, and the summation weights.

19. **Discrete Least Squares:** Suppose you are given the data points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \ldots, x_n\}$ and the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \ldots, f_n\}$, where $x_i > 0\ \forall i = 0, 1, 2, \ldots, n$.

    (a) For some reason, you think that $h(x) = a + bx + c\cos(x)$ is a great model for the data set. Find the best fit, in the least squares sense, for this model. **Find the normal equations.**

    (b) By a stroke of luck, it turns out that the basis functions $\{\Phi_0(x), \Phi_1(x), \Phi_2(x)\} = \{1, x, \cos(x)\}$ are orthogonal on the nodes $\vec{\mathbf{x}}$ with respect to summation against the weight function $w(x) = 1$. This should help you express the coefficients $\{a, b, c\}$ in a simpler way than in part (a).

20. **Discrete Least Squares Approximation:** Suppose you are given the data points $\vec{\mathbf{x}} = \{x_0, x_1, x_2, \ldots, x_n\}$ and the function values $\vec{\mathbf{f}} = \{f_0, f_1, f_2, \ldots, f_n\}$, where $x_i > 0\ \forall i = 0, 1, 2, \ldots, n$.

    (a) For some reason, you think that $h(x) = a + b\sqrt{x} + cx^3$ is a great model for the data set. Find the best fit, in the least squares sense, for this model. **Find the normal equations.**

    (b) Explain how the normal equations simplify if we have an orthogonal set of basis functions $\{\Phi_0(x), \Phi_1(x), \Phi_2(x)\}$, and we are trying to fit the model $g(x) = a\Phi_0(x) + b\Phi_1(x) + c\Phi_2(x)$ to the given data.

21. **The Fast Fourier Transform:** The bottlenose dolphin can generate sounds in the range from 250 Hz up to 150 kHz. The lower range, from 250 Hz up to approximately 40 kHz is used for social communication, and the upper range 40–150 kHz is primarily used for echo-location. The range for human hearing is from 20 Hz up to 20 kHz (on a very good day). On several web-sites on the net you can hear "dolphin sounds."

    Assume you have a waterproof microphone sophisticated enough to capture sounds in the bottlenose dolphin's range, an AD (Analog-Digital) converter that can sample the signal and save it to a data file that you can import into matlab. Think outside the box and device a method — using the FFT, IFFT, low/band/high-pass filters, clever heuristics and a DA (Digital-Analog) converter — so that you can place "dolphin sounds" (both social and echo-location) on your web-page as well.

22. **The Fast Fourier Transform:** In space, all alien species speak English(!?) Unfortunately, you have run into a species which communicates on a different frequency. Whereas the range for human hearing is from 20 Hz up to 20 kHz (on a very good day) and most human speech falls in the range from 300 Hz to 3400 Hz; the vocal range for this particular species is in the 90 kHz to 120 kHz band. Assume you have a access to a microphone sophisticated enough to capture sounds in the entire 20 Hz to 120 kHz regime, and an AD (Analog-Digital) converter that can sample the signal and save it to a data file that you can import into matlab. Think outside the box and device a method — using the FFT, IFFT, low/band/high-pass filters, clever heuristics and an additional DA (Digital-Analog) converter which accepts vector-input from matlab — so that you can have a conversation with this species.

23. **Polynomials:** Polynomials are cornerstones in most of our algorithms, therefore understanding the behavior of them is the key to understanding how these algorithms are derived, and how they perform.

    (a) Polynomials can be quickly (efficiently) evaluated using ***Horner's method***. Given a polynomial $P(x)$, and a point $x_0$, what does Horner's method give us? How is the result useful for polynomial root-finding $P(x) = 0$? (10 pts.)

    (b) One crucial use of polynomials is interpolation of given data points $\{(x_k, f_k)\}$, $k = 0, \ldots, n$.

       i. Write down the Lagrange building block $L_{n,k}(x)$ — that is, the $n$th degree polynomial which is one in $x_k$ and zero in $x_j$, $j \neq k$. (5 pts.)

       ii. Write down the Lagrange interpolating polynomial, which interpolates the data points $\{(x_k, f_k)\}$, $k = 0, \ldots, n$. (5 pts.)

(c) Interpolating polynomials are used to derive schemes for numerical computation of derivatives and integrals. It would seem that the more data points (samples) we have of the underlying function, the better.

i. In exactly ***one*** word, what is the problem with high-degree (interpolating) polynomials? (5 pts.)

ii. The error term for an $n$th degree interpolating polynomial is

$$\frac{f^{(n+1)}(\xi)}{(n+1)!} \underbrace{\prod_{k=0}^{n}(x-x_k)}_{m(x)}$$

where, in general, it is hard to say anything useful about $m(x)$. However, if we are free to select the points $x_k$ freely we can select an optimal (Chebyshev) placement of the points,

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right).$$

What can be said about $m(x)$ in this case? (5 pts.)

(d) Numerical integration schemes are derived by piecewise polynomial interpolation of a function. Unlike spline interpolation, for integration schemes we do not worry about the continuity of the derivatives at the points where sub-intervals meet.

i. If we base our numerical integration schemes on the use of interpolating polynomials using regularly spaced points, that is

$$\int_a^b f(x)\,dx \approx \int_a^b P_n(x)\,dx = \sum_{k=0}^{n} a_k f\left(a + \frac{k(b-a)}{n}\right)$$

then if $n$ is even, the error in the approximation (if the values $a_k$ are selected appropriately) is of the form

$$\mathcal{C}_n \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!}, \quad \mathcal{C}_n \in \mathbb{R}$$

***What is the highest degree polynomial for which this approximation scheme is exact? (The* <u>degree of precision.</u>*)*** (10 pts.)

ii. If we are free to place the points $x_k$ anywhere, then we can optimally place them and get the ***Gaussian quadrature formulas***. What is the degree of precision for the Gaussian quadrature scheme

$$\int_a^b f(x)\,dx \approx \sum_{k=0}^{n} a_k^* f\left(x_k^*\right)$$

which uses $(n+1)$ points ($n$ subintervals)? Here, $x_k^*$ and $a_k^*$ denote the Gaussian quadrature points, and the appropriate summation weights. (10 pts.)