CS 322: Problem Set 4

Due: Monday, July 22, 2002 (In Lecture)

The policies for this homework assignment are as follows:

- You must work by yourself on Question 1. You may work with at most one other person on Questions 2 and 3. Consult the course website for the Academic Integrity rules.
- Problem sets will be weighted equally in determining your final grade. The number of questions or total number of points on a given problem set is irrelevant.
- Writing will be graded on content, as well as on grammar, spelling, punctuation, etc.
- Mathematical exercises will be graded on the overall set-up of the problem as well as correctness.
- Points will be deducted on the Matlab questions for poorly commented code and inefficient code/redundant computations.
- For the Matlab questions, you should hand-in the appropriate plot(s) and the script/function(s) necessary to generate the plot(s).
- Submit your assignment in two different parts (Part 1 for Question 1 and Part 2 for Questions 2 and 3). Every student should submit Part 1. At the top of the first page for Part 1, write your name, the course number, the problem set number, Part 1, your e-mail address, your student ID number, and the date. Each team of up to two students should submit Part 2 jointly. The information at the top of Part 2 should be the same, except there should be up to two names at the top of the paper, and Part 1 should be replaced with Part 2.

1. (10 points)

   (a) Consider approximating f'(0.900), where $f(x) = \sin x$. The true value is $\cos 0.900 = 0.62161$. Use the following three-point centered-difference formula to approximate $f'(x)$:

   $$f'(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6} f^{(3)}(\xi),$$

   where ξ lies between $(x_0 - h)$ and $(x_0 + h)$ for stepsize h.

   You should compute quantities of the form $f(x_0 + k)$ to five digits past the decimal point.

   To help you with this, you should write a Matlab function Deriv3.m that has the following specification:

```
function fprime = Deriv3(fname,x0,h)
%
% Input:
% fname - a string containing the name of a function to differentiate
% x0 - the point at which to approximate f'(x)
% h - the stepsize
%
% Output:
% fprime is the approximation to f'(x0) using stepsize h and the
% 3-point centered difference formula.  Quantities of the form f(x0+k)
% should be computed using the feval command and rounded to 5 digits
% past the decimal point using the chop command.
```

   Write a script that uses Deriv3 and makes a table with the following three columns: h, approximation to $f'(0.900)$, and absolute value of the error. The table should be constructed for the following values of h : $0.001, 0.002, 0.005, 0.010, 0.020, 0.050$, and 0.100. Display your results to five digits to the right of the decimal place. Turn in a diary file with your results. Make a plot of the absolute value of the error vs. h.

Analyze the given centered-difference formula to determine an upper bound for the total error (i.e., truncation error plus rounding error) that occurs when this formula is used. Justify your choice for the upper bound for the roundoff error at each step. Determine the optimal choice for h using calculus to find the minimum of the error term. Is your answer consistent with the above plot?

(b) You learned in calculus that the derivative of a function f at x can be defined as:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}.$$

Write a Matlab function that computes approximates to $f'(x)$ using the following formula:

$$f'_n(x) = \frac{f(x+10^{-n})-f(x)}{10^{-n}}.$$

The function should have the following prototype:

```
function fprime = ApproxDeriv(fname,x,n)
%
% Input:
% fname - name of function to differentiate
% x - point to evaluate derivative approximation
% n - integer corresponding to iteration number
%
% Output:
% fn'(x) is the approximation to f'(x) according to the above formula.
```

Write a script that uses ApproxDeriv to compute approximations to $f'(x)$ for $f(x) = 3x^2 - 2sin(x)$ at $x = 2.1$ for $n = 1 : 1000$. Write a description of what happened and analyze why that particular behavior occurred.

2. (10 points) In lecture, we saw how composite quadrature rules could be used to compute

$$\int_a^b f(x)dx.$$

As it turns out, the same techniques can be modified to use in the approximation of multiple integrals. Consider the double integral given by

$$\int\int_R f(x,y)dA,$$

where $R = \{(x,y)|a \leq b, c \leq d\}$.

The first step in applying a composite quadrature rule is to divide R by partitioning both $[a,b]$ and $[c,d]$ into even numbers of subintervals. This creates evenly-spaced meshpoints in both directions. The second step is to consider the double integral as an iterated integral as follows:

$$\int\int_R f(x,y)dA = \int_a^b \left(\int_c^d f(x,y)dy\right) dx.$$

Next, we use the composite quadrature rule to evaluate

$$\int_c^d f(x,y)dy$$

by keeping x constant.

Finally, the composite quadrature rule is applied repeatedly to evaluate the outer integral.

Derive the Composite Simpson's Rule for the double integral above using this technique. You do not have to get an accurate expression for the error term. Simply label it ERROR.

Write a Matlab function called DoubleCompSimp.m that takes as input the name of a function in x and y, and constants a, b, c, and d and evaluates

$$Q = \int_a^b \left(\int_c^d f(x,y)dy \right) dx$$

using the Composite Simpson's Rule. Your function should return Q. Do not use any Matlab functions that perform numerical quadrature from your textbook; I am interested in having you write your own code. Make your code as efficient as possible.

Test your function by writing a Matlab script that calls DoubleCompSimp.m to evaluate

$$\int_{1.4}^{2.0} \int_{1.0}^{1.5} \ln(x+2y)dydx$$

using 4 subintervals in the x direction and 2 subintervals in the y direction. [Note that 4 and 2 correspond to n in the language of $Q_{NC(m)}^{(n)}$. Knowing that we're using the Composite Simpson's Rule specifies the value of m.]

3. (10 points) Thus far we have considered only deterministic means to approximating integrals. In this problem, we introduce a probabilistic approach called the Monte Carlo Method which can be used to approximate many quantities, one type being integrals.

   We now demonstrate how to use the Monte Carlo Method to approximate the value of the integral

$$\int_0^1 x^2 dx.$$

   The first step is to graph the function $y = x^2$ in the unit square. Second, we throw n darts at the square and keep track of where each of the darts lands. (Computationally this is done by generating random numbers that correspond to (x, y) values.) Third, we compute the ratio of the number of darts that lands below the curve to the number of darts that lands above the curve. Finally, we multiply our percentage by the area of the region we threw the darts at, which is 1. As the number of trials increases, this should be an increasingly good approximation to

$$\int_0^1 x^2 dx.$$

   This technique can be extended in the obvious way to compute double integrals.

   Consider the following problem:

   A machine in a factory attempts to produce widgets of length 10 inches having diameter 5 inches. However, there is a slight variation in the length and diameter of the components. A widget is considered usable if its length and diameter differ from the target values by less than 0.1 in. Let x denote the length and y denote the diameter in inches. Define $p(x, y)$ the probability density function for the variation in x and y as

$$p(x, y) = \frac{50\sqrt{2}}{\pi} e^{-100(x-10)^2} e^{-50(y-5)^2}.$$

What is the probability that a widget will be deemed usable?

In order to solve this problem, first formulate a double integral of $p(x, y)$ that corresponds to the probability that the widget is usable. Make use of the following fact: the probability that a widget has length between a and b and diameter between c and d = volume under the graph of p over the rectangle $a \leq x \leq b, c \leq y \leq d$.

Then, write a Matlab script called DoubleMonteCarlo.m that will estimate the probability using the Monte Carlo Method. Seed the random number generator using your favorite positive integer. See "help rand" to learn how to do this. Run your script for various numbers of trials and turn in your best estimate; show the output in a diary file.