CS 222: Introduction to Scientific Computing
Spring 2001
Problem Set 2

Handed out: Wed., Feb. 14.

Due: Wed., Feb. 21 in lecture.

The policies for this (and other problem sets) are as follows:

- You should hand in your on-time problem set in lecture in the box at the front of the room on the day it is due. Problem sets handed in elsewhere (TA's office, Upson 303, etc.) will be considered late. See the next bullet.
- Late papers may be handed in up to 24 hours late. For instance, this problem set may be handed in up to 11:00 am on Feb. 22. You can hand in a late paper in Upson 303. Late papers get an automatic deduction of 10%. The full late penalty is applied even if you turn in part of the solution on time.
- Problem sets may be done individually or in teams of two. Put your name or names on the front page. Re-read the academic integrity statement on the web for the policy concerning working in larger groups.
- Problem sets count for 20% of the final course grade. The lowest scoring problem set will be dropped.
- You need Matlab for some of the questions. Matlab is available in the following CIT labs: Upson and Carpenter.
- If you need clarification for a homework question, please either ask your question in section, lecture, or office hours or else post it to the newsgroup cornell.class.cs222. The professor reads this newsgroup and will post an answer.
- Write your section number (like this: "Section 2") at the top of the front page of your paper, and circle it. This is the section where your graded paper will be returned. As a reminder, Section 1 is Th 12:20, Section 2 is Th 3:35, Section 3 is F 2:30, Section 4 is F 3:35.

1. Download the following m-files from the textbook: CubicSpline.m, pwCEval.m, Locate.m. They are posted on the course home page.

   Use these m-files to answer the following question. Interpolate a not-a-knot cubic spline through the function $f(x) = \sqrt{1-x^2}$ at six evenly spaced breakpoints in $[-1, 1]$, i.e., at breakpoints $[-1; -.6; -.2; .2; .6; 1]$ (using CubicSpline). Then evaluate the difference between the true function and the interpolant at 100 evenly spaced x-coordinates in the same interval (using pwCEval to evaluate the spline), and print out the maximum value of the absolute difference.

   Now repeat all of this for the function $g(x) = \sqrt{1.2-x^2}$ on the same interval.

   You should discover that the interpolant of $g(x)$ is much more accurate than the interpolant of $f(x)$. Write a few sentences or formulas to explain why there is such a

big difference in accuracy. [Hint for the explanation: consider M_4 mentioned on p. 129 (p. 118 of the 1st edition) of the text.]

Hand in: a listing of your m-file, two plots, each one showing the true function and interpolant on the same axes (i.e., two plots each one analogous to the plot you made for Q3 of PS1), a printout of the two numerical differences in the two test cases, and the explanation requested in the preceding paragraph.

2. The following Matlab function is prone to overflow and also is not vectorized. Rewrite it (i.e., write a mathematically equivalent function) that is less prone to overflow and is fully vectorized.

   For example, if any entry of v exceeds 710, then then the function below overflows even though the final answer is not so large.

   Note: you do not have to implement your new function—you can just write it on paper. But please use correct notation. A hint for the dealing with overflow problem: consider scaling v.

```
function q = log_sum_exp(v)
% input v is an n-vector.
% output q is the log of the sum of the exponentials of
% entries.
n = length(v);
s = 0.0;
for j = 1 : n
  s = s + exp(v(j));
end
q = log(s);
```

3. Consider the polynomial $p(x) = (x - 3.5)^{15}$. Write a script in Matlab that plots this function (by evaluating it at closely spaced points) for the domain $[3, 4]$. Then multiply the factors of $p(x)$ together to obtain the array of coefficients in the standard-form representation $a_0 + a_1 x + \cdots + a_{20} x^{15}$ of $p(x)$ and evaluate the expanded form of the polynomial again over the interval $[3, 4]$ and plot this. You should notice a substantial difference between the two plots. The reason for the difference is that the second plot suffers from cancellation error. Why? (Hint: look at the coefficients.)

   Note: you can generate the standard-form polynomial of p either by multiplying it out in a loop (for example, in the manner that you used for computing `q` in Q2 of PS1) or by using the Matlab built-in function `poly`. You can evaluate a standard-form polynomial using the built-in function `polyval`.

   Hand in: the two requested plots, a listing of the m-file script that generates the plots, and an explanation of the reasons for the inaccuracy of the second plot. Use vectorization.