CS412 **Assignment 4 (as of 17oct00)** Fall 2000

due date: 24 October 2000

STAPLE your homework. MARK your homework clearly with your NAME. In addition, write the first letter of your LAST NAME boldly into the upper left corner of the first page of your homework . For this to be of real help, the vertical edge of the upper left corner should be longer than the horizontal edge.

1. (10+10+5 points)

(a) Construct the cubic Hermite interpolant q to the function $f(x) = \sin(x)$ at the points $-\pi/2$ and $\pi/2$ (i.e., the unique cubic polynomial that interpolates this function at the sequence $(x_1, x_2, x_3, x_4) := (-\pi/2, -\pi/2, \pi/2, \pi/2)$). Specifically, construct (by hand) the appropriate divided difference table and then use some of its entries to write down a suitable Newton form of the polynomial q.

(b) Use the error formula for polynomial interpolation to obtain as small a bound as you can for the maximum error $\max_{|x|\le\pi/2} |f(x) - q(x)|$.

(c) Using `matlab`, compute $|f(x) - q(x)|$ for a fine sequence x of points in the interval $[-\pi/2 \,.\,.\, \pi/2]$, take the maximum of these values, and comment on the difference (if any) between it and the bound you obtained in (b).

2. (25 points) Test the correctness of the book's `function GLWeights` (get a copy from the the book's homepage as pointed to on the homepage for `CS412`; else pick up a slightly differently written copy directly from the homework page). Specifically, write (and run) a script that checks whether, for $1 < m < 7$, the weights and nodes returned by `[w,x] = GLWeights(m)` really give a rule of order $2m$. (Theory tells us that a rule involving m nodes cannot have order greater than $2m$, so, all you have to check is whether this rule is exact for all polynomials of degree $< 2m$. Since any rule acts on functions **linearly**, i.e., $Q(\alpha f + \beta g) = \alpha Qf + \beta Qg$ for any rule Q, functions f, g and scalars α, β, this exactness only has to be checked for a basis, e.g., for the specific functions $f(x) = x^{j-1}$, $j = 1{:}2m$. Because of round-off, the computed rules may not get it exactly right, so you may have to choose a *reasonable(!)* tolerance when judging the results.)

Discuss anything unusual you might encounter.

3. (20) Write a `matlab` function `ppder(pp)` that starts off `function dpp = ppder(pp)` and produces the pp representation of g' from the pp-representation of g contained in `pp`. (Be sure not to use `for`-loops but, rather, use statements of the form `A.*repmat(w,size(A,1),1)` (for appropriate choices of `A` and `w`) to generate the coefficient array for g' from that for g.)

Use your `ppder` (twice) to compute the second derivative of the cubic spline interpolant to $f := \sin$ at the points in `x = linspace(0,2*pi,50)` (as supplied by the `matlab` command `spline(x, sin(x))` (be sure to do a `help spline`)) and compare it to the piecewise linear interpolant to $f'' = -\sin$ at the same mesh, `x`. Specifically, compute the maximum difference between these two piecewise linears (or, broken lines), and compute a numerical estimate of their error as an approximation to $-\sin$ (checking the maximum absolute error on a finer mesh).

3. (2*5 points) For each of the following, state whether it is true or false, and give a brief reason (or evidence) for your answer.

1. If all the n points in the Newton formula for the interpolating polynomial at those n points are the same, then the formula gives the Taylor polynomial of degree $< n$ at that point.
2. If `A` is a matrix of size `[m,n]`, then `reshape(A,n,m)` gives the transpose of `A`.
3. There exists a one-point quadrature rule that is exact for all polynomials of degree ≤ 2.
4. If `a` is a vector, then the `matlab` command `a(find(a<=0))=[]` removes from `a` all entries that are not positive.
5. The cubic spline interpolant with the not-a-knot end conditions at the points `(0:4)*pi` to the function $f(x) = 3x^3 - 2x^2 + x - \pi$ is f itself.