CS412 Spring Semester 2012

Homework Assignment #3

Due Thursday April 12th 2012, in class

Sum of all problems : 120%, Maximum possible score : 100%.

1. In this problem you will be exposed to `MATLAB`'s implementation of standard cubic splines (generated by the function `spline`) and Hermite splines (generated by the function `pchip`). You will also experimentally assess their accuracy, i.e. how rapidly they approximate a "real" function using interpolated sample points, as the distance between such sample points decreases.

   (a) [40%] In this part of the problem you will consider a function $f(x)$, and collect a number of sample points $(x_i, y_i), i = 0, 1, 2, \ldots, N$ from its plot (where $y_i = f(x_i)$). You will subsequently create a spline curve that interpolates all these sample points, and compare the result with the original function $f(x)$.

   For simplicity, we will focus on the specific function defined as

   $$f(x) = e^{-x^2}$$

   over the interval $[a, b] = [-3, 3]$.

   We proceed to generate a number of $N + 1$ equally spaced values $x_0 = a, x_1, x_2, \ldots, x_N = b$ such that

   $$x_k = a + kh$$

   where $N = (b - a)/h$ and the distance h between samples is a user-specified variable. In `MATLAB` we can generate a vector `x` containing all the data points $x_0 \ldots x_N$ as

   ```
   x=-3:h:3
   ```

   where, again, `h` is a variable that we have previously initialized to the desired value. Let `y` be the vector that contains the corresponding values of $f(x)$ at the corresponding locations (you should know how to construct this `y`). We can then compute a spline that interpolates all points (x_i, y_i) as

   `pp=spline(x,y)` or `pp=pchip(x,y)`

   Remember that the first expression will generate a standard cubic spline, using the not-a-knot formulation. The second expression is `MATLAB`'s version of cubic Hermite splines; the main difference from

the version discussed in class is that MATLAB's `pchip` does not explicitly ask for derivative values at x_i locations, but instead approximates their values from the point locations given as input.

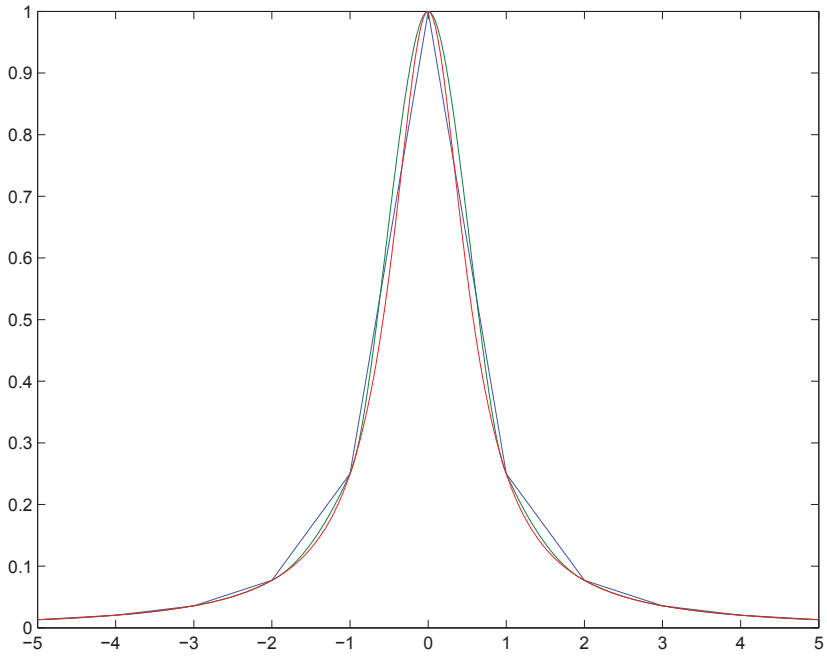
Regardless of the spline variant chosen, we can now plot the smooth interpolant. In order to do this, we need to evaluate the spline curve at a denser set of locations `xi`. For example, if we want to generate one hundred more points, we can write

```
xi=-3:0.01*h:3
```

After this step, use the MATLAB function `ppval` to compute a vector `xi` corresponding to the y-values of the spline interpolant at the (dense) points `xi`. In addition, generate another vector `ye` (with the same length as `xi` and `yi`) containing the *exact* values of $f(x)$ at the locations in `xi`, that is

$$\texttt{ye[k]} = f\left(\texttt{xi[k]}\right)$$

The plot below, which was generated by `plot(x,y,xi,yi,xi,ye)` gives an example of this process for similar but slightly different $f(x)$.



The blue curve indicates the sparse points (x_i, y_i) that the spline curve was generated to interpolate, the green curve is the spline interpolant, and the red curve is the "real" function f(x).

For this problem, use `MATLAB` to produce 6 plots like the one given, for all combinations of the following interval sizes $h \in \{1, 0.5, 0.25\}$ with the two possible spline implementations (`spline` vs. `pchip`). Turn in the generated plots, and the code used to produce them. What do you observe in terms of accuracy?

2. [40%] We say that the interpolation error of a spline method has order equal to d, if the discrepancy between the spline curve $s(x)$ and the "real" function $f(x)$ being approximated satisfies:

$$|f(x) - s(x)| \leq C \cdot h^d$$

In class we gave some bounds of this sort, which can be proven by theoretical means. However, it is also possible to approximate the exponent d (i.e. the order of the error) experimentally: In general we expect the following to hold

$$\max_{x \in [a,b]} |f(x) - s(x)| \approx C \cdot h^d$$

The maximum discrepancy between $f(x)$ and $s(x)$ can then be approximated by the *maximum distance between any two corresponding entries* of the vectors `ye` and `yi`; you should recognize that this is equal to the infinity-norm of the difference of the two vectors $\|\texttt{ye} - \texttt{yi}\|_\infty$, which is computed in `MATLAB`'s terms as

```
discrepancy=norm(xe-xi,inf)
```

This discrepancy will depend on the spline method used (`spline` vs. `pchip`) and on the value of h. In fact, since `discrepancy` $\approx C \cdot h^d$, we would expect to see the following behavior as we shrink the sample interval from h to $h/2$

- If d=1, the discrepancy is roughly reduced by a factor of 1/2.
- If d=2, the discrepancy is roughly reduced by a factor of 1/4.
- If d=3, the discrepancy is roughly reduced by a factor of 1/8.
- If d=4, the discrepancy is roughly reduced by a factor of 1/16, etc.

For the two spline methods described above, compute the discrepancy resulting from using values of $h \in \{1, 1/2, 1/4, 1/8, 1/16, 1/32\}$ (report the discrepancy values per method in a table, for each value of h). Can you infer what the order of error (d) is from these observations, for each of the two spline methods?

3. [40%] Prove the following properties (on paper):

   (a) Show that for any vector $\mathbf{x} \in \mathbf{R}^n$, the following inequalities hold:

   $$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n\|\mathbf{x}\|_\infty$$

   $$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty$$

   (b) Assume that positive constants c_1, c_2 exist, such that for any $\mathbf{x} \in \mathbf{R}^n$

   $$c_1\|\mathbf{x}\|_a \leq \|\mathbf{x}\|_b \leq c_2\|\mathbf{x}\|_a$$

   Here, $\|\cdot\|_a$ and $\|\cdot\|_b$ are simply two different vector norms. Show that in this case, we can also find positive constants d_1, d_2 such that

   $$d_1\|\mathbf{M}\|_a \leq \|\mathbf{M}\|_b \leq d_2\|\mathbf{M}\|_a$$

   for any *matrix* $\mathbf{M} \in \mathbf{R}^{n\times n}$. The norms in the last expression are the matrix norms induced from the respective vector norms.