CS 322: Assignment 1

Due: Monday, February 9, 2004, 4pm.

Do not submit work unless you have adhered to the principles of academic integrity as descibed on the course website:

`http://www.cs.cornell.edu/Courses/cs322/2004sp/`

Points will be deducted for poorly commented code, redundant computation that seriously effects efficiency, and failure to use features of MATLAB that are part of the course syllabus. In particular, use vector operations whenever possible. Pay attention to the course website for news that relates to this assignment.

Problem A (5 pts) Monte Carlo

This problem uses the Monte Carlo technique to estimate the area of a set in $\mathbb{R}^2$. For background, study the script `Darts` on page 37 in SCMV and the surrounding text that discusses the Monte Carlo idea. The key notion behind `Darts` is the "multiple" dart throw. If `N` is a positive integer and

```
x = -1 + 2*rand(N,1); y = -1 + 2*rand(N,1)
```

then we can think of the points `(x(i),y(i))`, `i=1:N` as the locations of `N` randomly thrown darts that land inside the square S defined by vertices $(1,1)$, $(-1,1)$, $(-1,-1)$, and $(1,-1)$. In this problem you are write a function `Area(M,N)` that estimates the area of the set

$$T \;=\; \{\ (x,y) \mid (x,y) \text{ is closer to } (0,0) \text{ than to the boundary of } S.\ \}$$

using the Monte Carlo method. Your implementation of `Area(M,N)` must have the following properties:

- It should base the area estimate on `M` multiple dart throws, each involving `N` darts. (In `Darts`, `M = 500` and `N = 100`.)
- It should produce a single figure that depicts the boundary of S and all the darts that land inside T. Regarding the latter, if vectors `u` and `v` house the coordinates of dart throws that you want to depict, then `plot(u,v,'.','markersize',1)` will display suitably sized dots, one for each dart. (Feel free to play with the marker size in order to produce a nice picture.)
- It should have no nested loops. This forces you to vectorize the processing of each multiple dart throw. You'll want to review how MATLAB handles boolean operations on vectors. Check out the built-in function `find`.
- It should have a title that displays the area estimate (use `8.5f` format) and the total number of throws, i.e, `N*M`.
- It should include the commands `axis equal` to ensure equal scales in the x and y directions, `axis off` to suppress the display of the axes, and `axis([-1.1 1.1 -1.1 1.1)` to pull back the target S from the edge of the window.

Submit a listing of `Area` and the figure produced by `Area(100,1000)`.

Problem B (5 pts) Fern Trajectories

In this problem you are to implement a function `newfinitefern(n)` that solves problem 1.17 in NCM. Take a look at the NCM script `finitefern(n)` and the associated discussion of `fern` in §1.3. Your implementation of `newfinitefern` must have these properties:

- It should produce in a single figure the fern associated with `finitefern(n)` and the four trajectories that are described in problem 1.17.
- If `u` and `v` are length `n` vectors that depict a trajectory, then it should be displayed with

  ```
  plot(u,v,'r',u(n),v(n),'*r')
  ```

  It should also display the coordinates of each trajectory limit point using

  ```
  gtext(sprintf('(%6.3f,%6.3f)',u(n),v(n)))
  ```

  (The built-in function `gtext` permits you to locate specified text in the current figure.)
- It should indicate the value of `n` in the title of the figure.

Submit a listing of `newfinitefern` and the figure produced by `newfinitefern(50000)`.

Problem C (5 pts) Derivative Approximation

Suppose f is a function that is defined everywhere and that all its derivatives exist. This problem is about three different methods for approximating the derivative of f at a point $x = a$. Two of the methods are given. You derive the third method. The error of the three methods is explored with a simple example.

We start with the Taylor expansion

$$f(a+h) \;=\; f(a) + f^{(1)}(a)h + O(h^2).$$

The $O(h^2)$ term actually has the form $f^{(2)}(\eta)h^2/2$. However, it is handier in this problem to designate these remainders with the big-Oh notation since we will be primarily interested in how various errors behave as a function of h. By rearranging the above equation we obtain

$$\texttt{Meth1}(a,h) \;=\; \frac{f(a+h)-f(a)}{h} \;=\; f^{(1)}(a) \;+\; O(h).$$

We can actually do better by sampling f at $a+h$ and $a-h$. Since

$$f(a+h) = f(a) \;+\; f^{(1)}(a)h \;+\; f^{(2)}(a)\frac{h^2}{2} \;+\; O(h^3)$$

we have

$$D_h(a) \;\equiv\; \frac{f(a+h)-f(a)}{h} \;=\; f^{(1)}(a) \;+\; f^{(2)}(a)\frac{h}{2} \;+\; O(h^2)$$

By setting h to $-h$ in this expression we get

$$D_{-h}(a) \;\equiv\; \frac{f(a-h)-f(a)}{-h} \;=\; f^{(1)}(a) \;-\; f^{(2)}(a)\frac{h}{2} \;+\; O(h^2)$$

If α and β are scalars then

$$\alpha D_{-h}(a) + \beta D_h(a) = (\alpha+\beta)f^{(1)}(a) + (\alpha-\beta)f^{(2)}(a)\frac{h}{2} \;+\; O(h^2)$$

If $\alpha+\beta = 1$ and $\alpha-\beta = 0$, then the lefthand side is an $O(h^2)$ approximation to $f^{(1)}(a)$. This can be done by setting $\alpha = \beta = 1/2$ giving:

$$\texttt{Meth2}(a,h) \;=\; \frac{f(a+h)-f(a-h)}{2h} \;=\; f^{(1)}(a) \;+\; O(h^2)$$

For small h, it is generally better to have $O(h^2)$ error than $O(h)$ error. Notice that `Meth2` is obtained by cleverly combining $D_h(a)$ and $D_{-h}(a)$, a pair of derivative approximations that have $O(h)$ error.

Use this same methodology to develop an $O(h^4)$ method. Start with this Taylor expansion of f about $x = a$:

$$f(a+s) = f(a) + f^{(1)}(a)s + f^{(2)}(a)\frac{s^2}{2} + f^{(3)}(a)\frac{s^3}{6} + f^{(4)}(a)\frac{s^4}{24} + O(s^5).$$

Thus,

$$D_s(a) \;\equiv\; \frac{f(a+s)-f(a)}{s} \;=\; f^{(1)}(a) \;+\; f^{(2)}(a)\frac{s}{2} \;+\; f^{(3)}(a)\frac{s^2}{6} \;+\; f^{(4)}(a)\frac{s^3}{24} \;+\; O(s^4).$$

Now determine scalars α, β, γ, and δ so that if

$$\texttt{Meth3}(a,h) \;=\; \alpha D_{-2h}(a) \;+\; \beta D_{-h}(a) \;+\; \gamma D_h(a) \;+\; \delta D_{2h}(a)$$

then

$$\texttt{Meth3}(a,h) \;=\; f^{(1)}(a) \;+\; O(h^4)$$

Thus, the idea is to cleverly determine a combination of the $O(h)$ approximations $D_{-2h}(a)$, $D_{-h}(a)$, $D_h(a)$, and $D_{2h}(a)$ so that the error is $O(h^4)$. Encapsulate all of these ideas by implementing the following MATLAB function:

```
  function [dfA,dfB,dfC] = AccuDerivative(fName,a,h)
% fName is a string that names an available function f that is defined everywhere.
% a and h are real scalars. Three approximations to f'(a) are returned:
% dfA = Meth1(a,h), dfB = Meth2(a,h), and dfC = Meth3(a,h).
```

Test your implementation by writing a script `A1C` that uses `AccuDerivative` to compute approximations to the derivative of $\sin(x)$ at $x = 1$. Your script must have the following properties:

- If `h = logspace(-16,-1,16)`, then for `k=1:16` it stores the absolute error of `Meth1(1,h(k))`, `Meth2(1,h(k))`, and `Meth3(1,h(k))` in `err1(k)`, `err2(k)`, and `err3(k)` respectively.
- It plots all of these vectors against `h` in a single window using `loglog`.
- It suitably labels the axes and includes a legend so that the three graphs can be identified.
- It includes a brief comment that explains the shape of the three graphs. (Hint: Read §1.5.2 in SCMV and consider the minimization of $h^d + \texttt{eps}/h$ for $d = 1, 2, 4$.)

Submit a listing of `AccuDerivative` and `A1C` together with the figure that is produced when `A1C` is run.

Problem D (5 pts) Pade Approximation

The Pade approximations to the exponential function e^x are described in §1.6.2 in SCMV. Read this section and become familiar with the SCMV functions `PadeCoeff` and `PadeArray`. (You'll have to understand `struct` and `cell`.) To illustrate the syntax, the script

```
x = 1;
P = PadeArray(3,3);
% P{i,j}.num is a row vector of length i
% P{i,j}.den is a row vector of length j
% Their components are accessed in the usual way with parens (not square brackets as
% mistakenly shown in the PadeCoeff comments.)
a0 = P{2,3}.num(1); a1 = P{2,3}.num(2);
b0 = P{2,3}.den(1); b1 = P{2,3}.den(2); b2 = P{2,3}.den(3);
f = (a0 + a1*x)/(b0 + b1*x + b2*x^2);
```

assigns to `f` the value of $R_{12}(x)$ at $x = 1$. Note that

$$R_{12}(x) = \frac{\texttt{a0} + \texttt{a1} \cdot x}{\texttt{b0} + \texttt{b1} \cdot x + \texttt{b2} \cdot x^2} = \frac{1 + \frac{1}{3}x}{1 - \frac{2}{3}x + \frac{1}{6}x^2}.$$

Write a script `A1D` that displays the absolute error of $R_{pq}(x)$ across the interval $[-1, 1]$ for nine instances defined by $p = 1{:}3$, $q = 1{:}3$. Your script must have the following properties:

- It uses `subplot` so that all nine graphs are displayed in a single figure.
- It uses `semilogy` for each plot. The absolute error *plus* `eps` should be displayed in each subwindow. (The "`eps`" ensures that we don't ever take the log of zero.)
- It examines the error at `x = linspace(-1,1,200)`.
- The evaluations of R_{pq} must be vectorized–no 1:200 loops.
- The value of p and q should be displayed in the title of each subwindow.

Submit a listing of `A1D` and a copy of the figure that it produces.