CAAM 453: Numerical Analysis I

Problem Set 6

Due: November 7th 2014 by 9am (beginning of class).

Note: All MATLAB *functions mentioned in this homework assignment can be found on the CAAM453 homepage, or come with* MATLAB . *You can use the* MATLAB *codes posted on the CAAM453 web-page. If you modify these codes, please turn in the modified code. Otherwise you do not have to turn in printouts of the codes posted on the CAAM453 web-page. Turn in all* MATLAB *code that you have written and turn in all output generated by your* MATLAB *functions/scripts.* MATLAB *functions/scripts must be commented, output must be formatted nicely, and plots must be labeled.*

Problem 1 (20 points)

In most floating point systems a reasonable approximation to machine precision ε_M can be obtained with the simple expression

$$\varepsilon_M \approx |3*(4/3-1)-1|$$

(a) In Matlab, compute

```
myeps = abs(3*(4/3 - 1) - 1);
format long e
eps_myeps = [eps ; myeps ]
```

Do the two values agree?

(b) Use your knowledge of correctly rounded floating point arithmetic (use the rules we discussed in class) to explain what you see.

(c) Explain why this approximation will not work when the base $\beta = 3$.

Problem 2 (40 points)

In this problem you will compare three different ways of computing an integral using quadrature. Newton-Cotes quadrature approximates the integral of a function by the integral of the degree-N polynomial that interpolates it at $N+1$ equispaced points. Clenshaw-Curtis quadrature approximates the integral of a function by the integral of the degree-N polynomial that interpolates it at $N+1$ Chebyshev points.

a. Write a function that computes the quadrature weights for Newton-Cotes and Clenshaw-Curtis. You may use MATLAB 's `poly`,`polyval`,`polyder`,`polyint`, and `roots`.

b. Use Newton-Cotes, Clenshaw-Curtis and the Composite Trapezoid rule approximate the integrals $\int_{-1}^{1} f(x)\mathrm{d}x$ for each of the following:

$$f(x) = e^{-x^2}, \qquad f(x) = (1+25x^2)^{-1}, \qquad f(x) = |x|.$$

In particular, for each f, produce a `semilogy` plot showing the number of points used in the approximation N versus the error between each of the three approximations and the true integral (whose values can be computed in MATLAB via `sqrt(pi)*erf(1)`, `2*atan(5)/5`, and `1`), for $N = 1, \ldots, 50$.

c. Why do the three different functions in part b produce such different results?

d. For the first function $f(x) = e^{-x^2}$, use MATLAB 's `tic` and `toc` commands to time how long it takes to compute the Composite-Trapezoid and Clenshaw-Curtis approximations for $N = 20$ (Include the time for computing the nodes and weights). Compare your times to the time required to integrate this same f using MATLAB 's all-purpose adaptive quadrature routione, `quad`, with precision `1e-15`.

Problem 3 (40 points) *(this problem is pledged)*

Write a MATLAB code that computes

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

in an adaptive manner (we discussed that in class).

The numerical quadrature algorithm that you have to program should satisfy user-specified precision requirements i.e.

$$\left| Q(f,a,b) - \int_a^b f(x)dx \right| < \varepsilon.$$

Moreover, the function should return the total number of evaluations of function f that have been done in the quadrature process. The calling sequence should be

```
[val,N]   =   myQuadrature(f,a,b,eps)

Input:

      f   is a handle to a function f(x)
      a   is the lower bound of the interval of integration
      b   is the upper bound of the interval of integration
```

```
   eps   is the user-defined precision

Output:

   val   is the computed value of the integral
     N   is the number of evaluations of function f
```

You can verify your results using build-in MATLAB routines `quad` or `quadl`.

Your should test your program with the following functions

```
[val1, N1] = myQuadrature ( inline('1+atan(30*x)','x'), -1, 1,1.e-6)
[val2, N2] = myQuadrature ( inline('x+sin(x.^4)','x'), 0, 3,1.e-12)
[val3, N3] = myQuadrature ( inline('exp(-x.^2)','x'), -100, 100,1.e-12)
```

Graders will test your program against other functions. You will be evaluated both on the exactness of your code (i.e. getting the answer with the right accuracy) and on its performance (i.e. minimizing the number of evaluations of f).

In addition to the usual written report, each student has to send an email to `remacle@rice.edu` with a single file containing his MATLAB program. The object of the mail has to be `CAAM453_HW6` and the program in attachment has to be named `YourFirstName_YourLastName.m`.