MATH 4093/5093 Homework 3 Due Fri, 09/17/10

Problem 1. The so-called *error function*, $\operatorname{erf} x$, is defined as the integral

$$\operatorname{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \ .$$

(a) What is the value of $\operatorname{erf}(0)$?

(b) Find the derivatives $\operatorname{erf}'(x)$, $\operatorname{erf}''(x)$, and $\operatorname{erf}'''(x)$.

(c) Write down the Taylor polynomial of degree 3, $T_3(x)$, of the function $\operatorname{erf}(x)$ about the point $x_0 = 0$.

(d) Use the Taylor polynomial obtained in (b) to find the approximate value of $\operatorname{erf}(0.2)$. The exact value of $\operatorname{erf}(0.2)$ is $0.222702589210478\ldots$; compute the numerical value of the absolute error, $|T_3(0.2) - \operatorname{erf}(0.2)|$, and the relative error in approximating $\operatorname{erf}(0.2)$ by $T_3(0.2)$.

Problem 2. The values of $\operatorname{erf}(x)$ can be obtained in different ways (all of them using numerical methods). The Taylor expansion of $\operatorname{erf}(x)$ around 0 will not be very accurate for relatively large values of the argument, say, for $x = 1$. The first method for computing $\operatorname{erf}(1)$ is simply to compute the value of the integral numerically. There are many methods for numerical computations of integrals, but, since we have not discussed them, here you will find $\operatorname{erf}(1)$ by solving an IVP for an ODE. The Matlab codes for solving IVPs are available at `http://www.pcs.cnu.edu/~bbradie/mivps.html`

(a) Write down a first-order ODE that $\operatorname{erf}(x)$ satisfies.

Hint: This question is equivalent to asking what the first derivative of $\operatorname{erf}(x)$ is.

(b) Think of an initial condition for $\operatorname{erf}(x)$. There is one value of x for which you know $\operatorname{erf}(x)$ exactly.

(c) Compute the value of $\operatorname{erf}(1)$ by integrating the IVP formulated in parts (a) and (b) using the Matlab code `mod_euler.m` (implementing the modified Euler method, which is a Runge-Kutta method of order 2). Do this for stepsizes $h = \frac{1}{10}$, $\frac{1}{100}$, $\frac{1}{1000}$ and $\frac{1}{10000}$. Write down the value of the absolute error for each h. What does the dependence of the absolute error on h seem to be? Does your empirical observation agree with what you expected?

Remark: The function $\operatorname{erf}(x)$ exists in Matlab – to get the value of $\operatorname{erf}(1)$, simply type `erf(1)`.

(d) Do the same as in part (c), but using the Matlab code `rk4.m` (which implements the classical Runge-Kutta method of order 4), and only for $N = 10$ and 100 (simply because for $N = 1000$ the numerical error will be of order the "machine epsilon"; you can find the machine epsilon in Matlab by typing `eps` and pressing ENTER). Again, discuss your findings about the error in the light of the theoretical predictions.

Problem 3. Let f be a function of two variables defines as $f(x, y) = \frac{\cos x}{y}$.

(a) Expand the function f in a Taylor polynomial around the point $(\frac{\pi}{3}, 5)$ keeping only the terms linear in the increments.

(b) Compute the approximate value of $f(\frac{\pi}{3} + 0.03, 5.1)$ coming from the Taylor polynomial of degree 1 obtained in part (a).

(c) Compute the exact value of $f(\frac{\pi}{3} + 0.03, 5.1)$, as well as the actual values of the absolute and the relative errors of the approximate value obtained in part (b).

Problem 4. In this problem you will study in detail the piecewise-linear interpolation of the function $f(x) = \frac{1}{x}$ on the interval $[1, 2]$, and then on the interval $[1, 4]$. Piecewise-linear interpolation of a function uses linear interpolation between each pair of consecutive values of the argument. For example, to find the piecewise-linear interpolant of $f(x) = \frac{1}{x}$ based on its values at $x = 1$, 2, and 4, you have to compute the Lagrange interpolation polynomial of degree 1 whose graph passes through the points $(1, f(1))$ and $(2, f(2))$, and the Lagrange interpolation polynomial of degree 1 whose graph passes through $(2, f(2))$ and $(4, f(4))$; these two Lagrange polynomials together constitute the desired piecewise-linear interpolant of the function $f(x) = \frac{1}{x}$. The graphs of the function $f(x) = 1/x$ and its piecewise-linear interpolant are shown in Figure 1.

(a) Find the first order Lagrange polynomial $P_1(x)$ of $f(x) = \frac{1}{x}$ that passes through the points $(1, f(1))$ and $(2, f(2))$.

(b) Let $E_{\text{true on } [1,2]} := \max_{x \in [1,2]} |f(x) - P_1(x)|$ be the *true* error of the first order Lagrange interpolation. Find the numerical value of $E_{\text{true on } [1,2]}$ "by paper and pencil".

Hint: You first have to find the value x^* of the argument that maximizes the expression $|f(x) - P_1(x)|$. Note that f is concave up, so that the graph of P_1 lies above the graph of f, therefore $|f(x) - P_1(x)| = P_1(x) - f(x)$.

(c) Find the rigorous upper bound $E_{\text{rigorous on } [1,2]}$ of the error of the linear interpolation on $[1, 2]$ given by the Theorem 3.3 on page 348 of the book. Note that you do not know the value of ξ in this bound, so you will have to take the maximum of the absolute value of the derivative in this bound over the whole interval $[1, 2]$. Separately, you will have to find the maximum value of the absolute value of the product of $(x - x_i)$

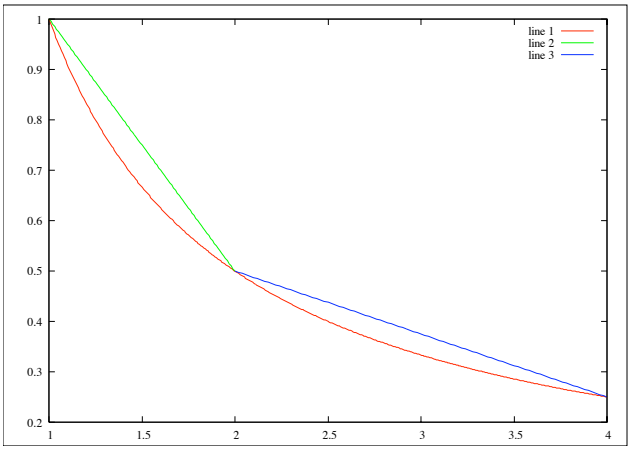

Figure 1: Piecewise-linear interpolation of $f(x) = 1/x$ on the interval $[1, 4]$ by using the values of $f(x)$ for $x = 1$, 2, and 4.

terms (look at the sign of each $(x - x_i)$ and get rid of the absolute values before taking derivatives). In other words, you have to find

$$E_{\text{rigorous on } [1,2]} = \frac{1}{(n+1)!} \max_{\xi \in [a,b]} |f^{n+1}(\xi)| \max_{x \in [a,b]} \prod_{k=0}^{n} |x - x_k| \ .$$

Find the exact value of this bound, and then compute its numerical value. Compare with the exact value of the error found in part (b).

(d) Now find the Lagrange interpolating polynomial of f over the interval $[2, 4]$, and write your results from parts (a) and (c) together in the form

$$P_{\text{piece-lin}}(x) = \begin{cases} b_1 x + c_1 \ , & x \in [1, 2] \ , \\ b_2 x + c_2 \ , & x \in [2, 4] \ . \end{cases}$$

Remark: It is easy to check your results: the piecewise-linear interpolant must be a linear function on $[1, 2]$ and $[2, 4]$ and must satisfy $P_{\text{piece-lin}}(1) = f(1)$, $P_{\text{piece-lin}}(2) = f(2)$, $P_{\text{piece-lin}}(4) = f(4)$.

(e) Use your result from part (d) to compute $P_{\text{piecewise-linear}}(1.25)$, and compare its value with $f(1.25)$.

Problem 5. This problem is a continuation of the previous one.

(a) One can use interpolants in approximate computations. For example, we can use the piecewise-linear interpolant of a function to obtain an approximate the integral of a function over an interval. Use the piecewise-linear interpolant $P_{\text{piece-lin}}(x)$ found in part (d) of the previous problem to approximate $\int_1^4 f(x)\,dx$ by $\int_1^4 P_{\text{piece-lin}}(x)\,dx$.

(b) In part (b) of the previous problem you found the rigorous upper bound $E_{\text{rigorous on [1,2]}}$ on the error in interpolating f by a linear function on $[1, 2]$. I computed the rigorous upper bound $E_{\text{rigorous on [2,4]}}$ on the error in interpolating f by a linear function on $[2, 4]$, and found that $E_{\text{rigorous on [2,4]}} = \frac{1}{8}$. Use the values of $E_{\text{rigorous on [1,2]}}$ and $E_{\text{rigorous on [2,4]}}$ to find a rigorous upper bound on the error $\left|\int_1^4 f(x)\,\mathrm{d}x - \int_1^4 P_{\text{piece-lin}}(x)\,\mathrm{d}x\right|$.

(c) Compute the true value of $\left|\int_1^4 f(x)\,\mathrm{d}x - \int_1^4 P_{\text{piece-lin}}(x)\,\mathrm{d}x\right|$ and compare it with the rigorous upper bound found in part (b).

An exercise in Matlab (<u>NOT to be turned in!</u>). To make Figure 1, I used Bradie's code `lagrange.m` (available at `http://www.pcs.cnu.edu/~bbradie/matlab.html` under *Interpolation*). If you know the values of the function f at the points x_0, x_1, ..., x_n, you can find the coefficients of the Lagrange polynomial $P_n(x)$ of order n that interpolates f as follows. Save the values of x_j as a vector `xx`, and the values of $f(x_j)$ as a vector `yy`. Then type, say, `lagr=lagrange(xx,yy)` and you will obtain the values of the coefficients of the Lagrange polynomial $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$. Suppose that you want to compute the Lagrange interpolating polynomial $P_2(x)$ of the function $f(x) = 1/x$ based on the values of $f(x)$ for $x = 1$, 2 4. Type `xx=[1 2 4]` , `yy=1.0 ./ xx` , and `lagr=lagrange(xx,yy)` , and this will define the vector `lagr` whose components are `lagr(1)` $= 0.125$, `lagr(2)` $= -0.875$, and `lagr(3)` $= 1.75$. This means that $P_2(x) = 0.125x^2 - 0.875x + 1.75$. To obtain, say, the value $P_2(3.1)$, you can type `polyval(lagr,3.1)` – see more about the Matlab command `polyval` in Section 11 of Ed Overman's *Matlab Overview* (available at the class web-site). I made Figure 1 by typing

```
x_dense = linspace( 1.0, 4.0, 401);
y_dense = 1.0 ./ x_dense;
plot(x_dense,y_dense)    % plotting y=1/x on [1,4]
hold on
xx = [1 2]
yy = 1.0 ./ xx
lagr = lagrange( xx, yy)
polyval( lagr, 1.0)     % just a test
polyval( lagr, 2.0)     % just a test
polyval( lagr, 1.5)     % just a test
x_dense = linspace( 1.0, 2.0, 101);
plot ( x_dense, polyval( lagr, x_dense)) % plotting the Lagr poly on [1,2]
xx = [2 4]
yy = 1.0 ./ xx
lagr = lagrange( xx, yy)
x_dense = linspace( 2.0, 4.0, 101);
plot ( x_dense, polyval( lagr, x_dense)) % plotting the Lagr poly on [2,4]
```

In Matlab there are more sophisticated commands to deal with plots, like `fplot`, `ezplot`, etc. (see Section 4.1 of Overman's text), but instead of Matlab I used Octave, where these commands are not available.