MATH 4093/5093 Homework 8 Due Fri, 4/15/2011

Problem 1. The so-called *error function*, $\mathrm{erf}(x)$, of great importance in probability, statistics and partial differential equations, is defined as

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

The integral in the definition of $\mathrm{erf}(x)$ cannot be expressed in terms of elementary functions, so that the values of $\mathrm{erf}(x)$ must be found using some numerical method.

One way to do this is to use the Taylor expansion of $\mathrm{erf}(x)$. For example, the Taylor polynomial of degree 7 around $x = 0$ of the function $\mathrm{erf}(x)$ is

$$T_7(x) = \frac{2x}{\sqrt{\pi}} - \frac{2x^3}{3\sqrt{\pi}} + \frac{x^5}{5\sqrt{\pi}} - \frac{x^7}{21\sqrt{\pi}} \ .$$

(Think how one can compute the derivatives of $\mathrm{erf}(x)$.) The value of $T_7(1)$ is $\frac{52}{35\sqrt{\pi}} \approx 0.8382245241280951$. Comparing this with the exact value, $\mathrm{erf}(1) \approx 0.8427007929497149$, we see that the relative error is about $0.53\,\%$. To achieve accuracy of about 10^{-16}, one has to use the Taylor polynomial $T_{34}(x)$ of $\mathrm{erf}(x)$ of degree 34, which is a daunting task (just try to obtain $T_{10}(x)$, to get an idea about the length of the calculations; no need to attach your calculations to the homework).

Another way to compute the values of the error function is to use numerical integration. This is easy and straightforward to do, but we have not discussed it in the class so far, so we won't pursue it here.

In this problem you will find the value of $\mathrm{erf}(1)$ by solving an appropriate IVP for an ODE. One can use any method for solving the IVP, but in this method you will use two Runge-Kutta methods, whose codes are available at the class web-site.

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

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

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

(c) Compute the value of $\mathrm{erf}(1)$ 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 (there is no need to attach your printout). 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 $\mathrm{erf}(x)$ exists in MATLAB – to get the value of $\mathrm{erf}(1)$, 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 of 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 2. 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.

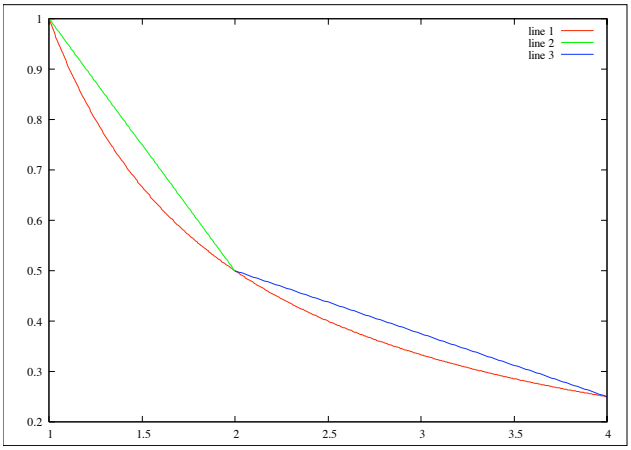

Figure 1: Piecewise-linear interpolation of $f(x) = 1/x$ on the interval $[1, 4]$ by using the values of $f(x)$ for $x = 1$, 2, and 4.

(a) Find the first-degree 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-degree 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 $E_{\text{rig, } [1,2]}$ 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 $(x - x_i)$ 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

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

Find the exact numerical value of this bound, and compare it 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 $P_{\text{piece-lin}}(1.25)$; compare it with $f(1.25)$.

Problem 3. 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 $\int_1^4 f(x)\,\mathrm{d}x$ by $\int_1^4 P_{\text{piece-lin}}(x)\,\mathrm{d}x$.

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

Hint: You can use the following standard mathematical tricks:

$$\left|\int_1^4 g(x)\,\mathrm{d}x\right| \le \int_1^4 |g(x)|\ \mathrm{d}x = \int_1^2 |g(x)|\ \mathrm{d}x + \int_2^4 |g(x)|\ \mathrm{d}x \ .$$

(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).

Problem 4. To make Figure 1, I used the MATLAB code `lagrange.m`, which in turn calls the MATLAB code `lagrange_poly.m` (both available at the class web-site). The code `lagrange.m` takes the values x_i and $y_i = f(x_i)$ (for $i = 0, 1, \ldots, n$), and returns the coefficients of the Lagrange polynomial P_n of degree n that interpolates f, i.e., such that

$$P_n(x_j) = f(x_j) \quad \text{for } j = 0, 1, \ldots, n \ .$$

The code `lagrange_poly.m` constructs the polynomial $L_{n,j}$ associated with the jth interpolating point (x_j, y_j) of a given set $\{(x_i, y_i)\}_{i=0}^{n}$ of interpolating points.

Here is how you use these two MATLAB codes. 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, and 4. Type

```
xx = [1 2 4]
yy = 1.0 ./ xx
lagr = lagrange(xx,yy)
```

(to recall what the command `./` does, type `help ./`). These commands define the vector `lagr` whose components are `lagr(1)` $= 0.125$, `lagr(2)` $= -0.875$, and `lagr(3)` $= 1.75$. This means that $P_2(x) = 0.125\, x^2 - 0.875\, x + 1.75$. To obtain, say, the value $P_2(3.1)$, you can type `polyval(lagr,3.1)` – see more about the 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,'r')    % 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),'g') % 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),'b') % 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 and MATLAB's help menu.

In this simple problem you have to use the codes `lagrange.m` and `lagrange_poly.m` in order to compute and then plot the interpolating polynomial of a certain set of points, which will

be a good illustration of the dangers of using polynomial interpolation with polynomials of high degrees. Consider the set of points $\{(x_i, y_i)\}_{i=0}^{11}$, where $x_i = i$ $(i = 0, 1, \ldots, 11)$, and

$$y_i = \begin{cases} x_i & \text{for } i = 0, 1, \ldots, 5 \ , \\ x_i + 1 & \text{for } i = 6, 7, \ldots, 11 \ . \end{cases}$$

Save the x-coordinates x_i in a vector `x`, and the y-coordinates y_i in a vector `y`. Then use the code `lagrange.m` to find the coefficients of the Lagrange interpolating polynomial, and finally plot the points $\{(x_i, y_i)\}_{i=0}^{11}$ (using circles) and the interpolating polynomial (using straight lines). Attach the printout of your MATLAB session and the plot of the points and the interpolating polynomial. Discuss briefly what you observe and what conclusions you can draw.

Problem 5. The MATLAB code `ab4.m` from the class web-site implements the 4-step 4th-order Adams-Bashforth method (AB4) given by

$$w_{i+1} = w_i + \frac{h}{24}\Big[55\, f(t_i, w_i) - 59\, f(t_{i-1}, w_{i-1}) + 37\, f(t_{i-2}, w_{i-2}) - 9\, f(t_{i-3}, w_{i-3})\Big] \ .$$

Modify this code to create the code `yourfirstname_yourfamilyname_pred_corr_4.m` that implements the predictor-corrector method based on the AB4 method above and the 3-step 4th-order Adams-Moulton method (AM3) given by

$$w_{i+1} = w_i + \frac{h}{24}\Big[9\, f(t_{i+1}, w_{i+1}) + 19\, f(t_i, w_i) - 5\, f(t_{i-1}, w_{i-1}) + \ f(t_{i-2}, w_{i-2})\Big] \ .$$

(a) Attach a printout of your MATLAB code and upload it in the Dropbox of D2L.

(b) Test your code on the initial-value problem

$$y'(t) = -y + \sin t \ , \quad t \in [\pi, 2\pi] \ ,$$
$$y(\pi) = 1 \ ,$$

whose exact solution is $y_{\text{exact}}(t) = \frac{1}{2}\,(\mathrm{e}^{\pi - t} + \sin t - \cos t)$. Compute the error $|y(2\pi) - y_{\text{exact}}(2\pi)|$ for $N = 10, 100, 1000$; here N is the number of intervals in which the interval $[\pi, 2\pi]$ is divided, i.e., the last argument of the function `ab4` (note that the arrays `wi` and `ti` created by `ab4.m` are of size $N + 1$). Attach a printout of your MATLAB session.