Homework set 1

Christian Huber (christian.huber@eas.gatech.edu), Carlos Cardelino (carlos.cardelino@eas.gatech.edu)

July 14, 2011

1 Problem 1 - Estimation of infinite series

(a) Write a Matlab script **ComputingNe.m** to compute the number *e* using a series approximation

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}. \tag{1}$$

Within the script, define the true (here approximated) value of *e* as 2.718281828459045 and stop the execution when the approximation reaches 2.71828182845. The output should display the approximated *e* and the number of terms (n) required in the series.

(b) Write a Matlab script **ComputingPi.m** to compute the number π using two different series approximations

$$\frac{\pi}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2} \tag{2}$$

$$\frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{2n}{2n-1}\frac{2n}{2n+1}. \tag{3}$$

The execution of the script should stop when the approximation of π reaches 3.1415926. For each series, show the result (approximate π value) obtained and the number of terms used in the series.

Turn copies of **ComputingNe.m** and **ComputingPi.m** and the 6 numerical values corresponding to the number approximations and number of terms in the series.

2 Problem 2 - Roots of a continuous function

(a) Write a Matlab function file **evaluate_NR.m** that takes one variable x as input and provides two scalar outputs $f(x)$ and $df/dx(x)$, where $f(x)$ is a continuous function defined as

$$f(x) = x^2 + x - 6. \tag{4}$$

(b) Write a Matlab script **NR.m** that implements the Newton-Raphson method discussed in class to solve for the roots of f(x). The script should ask for (1) an initial value x_0 as a starting point for the

iterative root-finding process and (2) a tolerance value ($Tol = 1e-6$) to stop the iterative process once an accurate estimation is calculated. Write a **while** loop that computes a new estimation of the root (starting with the initial input x_0) and that uses your function **evaluate_NR.m**

$$x_1 = x_0 - \frac{f(x_0)}{\frac{df}{dx}(x_0)}, \tag{5}$$

until

$$E = \left| \frac{x_{i+1} - x_i}{x_i} \right| < Tol. \tag{6}$$

(c) Start with $x_0 = 1.5$, provide the last estimation (within the tolerance Tol) and the number of iterations required to reach it. Repeat the same operation with $x_0 = -1$, what is the final value you obtain? How many iterations did it require? What happens if you start with $x_0 = 0.5$? Why should you be concerned?

Turn in copies of **evaluate_NR.m** and **NR.m** as well as the numerical values you obtained (2 values of x_{final} and two iteration counts), plus a discussion about the case $x_0 = 0.5$.

3 Problem 3 - Numerical integration

In this problem you will write 3 scripts: **midpoint.m**, **trapezoid.m** and **gauss_quad.m** to compute numerical approximation of definite integrals.

(a) Load the Matlab function **HW1_fct.m** which takes one variable x as input and computes

$$f(x) = x^3 + 2x^2 - 3. \tag{7}$$

(b) Write the script **midpoint.m**, which implements the midpoint rule (introduced in class) for numerical integration. The script should be designed to ask users for (1) the lower bound of the integral a, (2) the upper bound b, as well as (3) the number of segment N that will divide the interval $[ab]$. The integral estimation using the midpoint rule is defined as

$$I_m = h \sum_{i=1}^{N} f(x_{i-1/2}), \tag{8}$$

where $x_{i-1/2} = a + i * h/2$ and $h = (b-a)/N$.

(c) Write the script **trapezoid.m**, which implements the trapezoid rule (also introduced in class) for numerical integration. Similarly, the script should be designed to ask users for (1) the lower bound of the integral a, (2) the upper bound b, as well as (3) the number of segment N that will divide the interval $[ab]$. The integral estimation using the trapezoid rule is defined as

$$I_m = h \sum_{i=1}^{N} \frac{f(x_i) + f(x_{i+1}}{2}, \tag{9}$$

where $x_1 = a$, $x_i = a + (i-1)h$, $x_N = b$ and $h = (b-a)/N$.

order k	x_j, $j = 1, \ldots k$	w_j, $j = 1, \ldots k$
1	0	2
2	$\pm\frac{1}{\sqrt{3}}$	1
3	$0; \pm\sqrt{\frac{3}{5}}$	5/9
4	$\pm\sqrt{(3-2\sqrt{6/5})/7}; \pm\sqrt{(3+2\sqrt{6/5})/7}$	$(18+\sqrt{30})/36; (18-\sqrt{30})/36$

(d) Write the script **gauss_quad.m**, which implements a gaussian quadrature integration of order k (also introduced in class). The script should be designed to ask users for (1) the lower bound of the integral a, (2) the upper bound b, as well as (3) the order of the quadrature k. Write a function file **GQ.m** which accepts a single input variable k (the order) and sends back two arrays $\mathbf{x}$ and $\mathbf{w}$ which are respectively the position of the quadrature points and the individual weights (see table 1 for values). In function **GQ.m**, use the switch command to obtain the correct set of positions $\mathbf{x}$ and weights $\mathbf{w}$ when the command $QG(k)$ with $k = 1, \ldots 4$ is executed.

In the script **gauss_quad.m**, the integral is estimated with

$$I_G = \frac{(b-a)}{2} \sum_{j=1}^{k} w_j f\left(\frac{(b-a)}{2} x_j + \frac{(a+b)}{2}\right). \tag{10}$$

(e) Use the scripts **midpoint.m** and **trapezoid.m** to compute the integral of the function $f(x)$ in **HW1_fct.m** over the interval $a = 1$ and $b = 10$ using $N = 1, 2, 4, 6, 10 and 20$, write down the values of $I_{m,N}$ and $I_{t,N}$, where N is the number of points where $f(x)$ was estimated during the computation of the integral. Use the same function **HW1_fct.m** and compute the integral over the same range (a, b) with the gaussian quadrature script for order $k = 1, 2, 3, 4$. Write down these values and discuss the convergence of the gaussian quadrature that uses only 4 $f(x)$ ($k = 4$) compared to $N = 4$ with the midpoint or trapezoid rules.

(f) Plot the values obtained above for $I_{m,N}$ and $I_{t,N}$ as function of N on the same loglog plot (use legends and labels for both x and y axes), add the power-law $O(N) = N^{-2}$ on the same plot to show the slope that would be expected from second order methods. What do you observe ?

(g) **BONUS**: Replace the function $f(x) = x^3 + 2x^2 - 3$ with $f(x) = \mathrm{erfc}(x)$ (the complementary error function). Integrate $f(x)$ from $a = 0$ to $b = 10$ with $N = 4, 10, 100, 500$ with the midpoint and trapezoid rule scripts. Write down the results, compare them to a fourth order gaussian quadrature ($k = 4$). What do you observe ? The analytical solution should slowly converge close to $1/\sqrt{\pi}$. Why is the gaussian quadrature performing so badly compared to the two other methods?

Turn back your results and discussions (when discussions are requested) and copies of each script, function files.