MATH 4073 **Homework 2** **Due Wed, 9/4/13**

Problem 1. Even simple tasks like solving a quadratic equation numerically should be approached with care. One can just use the formula for the roots of a quadratic equation

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \ .$$

– the MATLAB function `nikola_petrov_quadr_eqn1.m` (available at the class web-site) does exactly this. MATLAB works with complex numbers – to see this, solve the quadratic equation $x^2 - 6x + 25 = 0$ by typing `nikola_petrov_quadr_eqn1(1,-6,25)`.

(a) Use `nikola_petrov_quadr_eqn1.m` to solve the quadratic equation

$$10^{-5}\, x^2 - 500000\, x + 1 = 0 \ . \tag{1}$$

Compute the absolute and the relative error of the values this function gives for the roots of the equations. The exact roots of this equation (with 20 digits of accuracy) are

$$\begin{aligned} x_1 &= 2.0000000000000000800\ldots \times 10^{-6} \ , \\ x_2 &= 5.0000000000000002000\ldots \times 10^{10} \ . \end{aligned}$$

Remark: To get 10^{-5} in MATLAB, type `1e-5` (*not* `10e-5` , as sometimes people do).

(b) The loss of accuracy in the computations in part (a) was due to the fact that in computing

$$500000 - \sqrt{500000^2 - 4 \times 10^{-5} \times 1}$$

we subtract two nearly equal numbers, which leads to a loss of accurate digits. On the other hand, computing

$$500000 + \sqrt{500000^2 - 4 \times 10^{-5} \times 1}$$

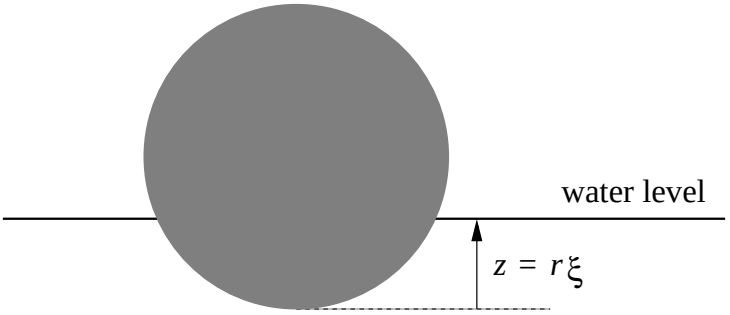
is perfectly safe numerically. To avoid the numerical problems caused by subtracting nearly equal numbers, one can use the "standard" formula when there is no danger, while in the case when the sign of the b is such that a dangerous cancellation may occur, the expression for the root can be rewritten as

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} = \frac{4ac}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \ . \tag{2}$$

If the computation of $-b - \sqrt{b^2 - 4ac}$ is dangerous numerically, then the computation of $-b + \sqrt{b^2 - 4ac}$ will be perfectly safe.

Write a MATLAB function called `yourfirstname_yourfamilyname_quadr_eqn2.m` that uses the "standard" formula for one of the roots and (2) for the other root (you have to use the MATLAB command `if` in order to check the sign of b). Use your function `yourfirstname_yourfamilyname_quadr_eqn2.m` to solve the quadratic equation (1). Attach a printout of your program and the result of running it to solve the quadratic equation (1).

Problem 2. A solid ball of radius r is floating in water, as shown in Figure 1.



The density of the ball is equal to ρ_{ball}, and the density of water is ρ_{water}. The ball is lighter than water, so it is floating in it; let z be the vertical distance from the lowest point of the ball to the surface of the water, as shown in the figure.

Using elementary integration, one can show that the part of the sphere submerged in the water has volume

$$V_{\text{submerged}} = \frac{\pi}{3}\, z^2\, (3r - z) \tag{3}$$

(you do *not* need to prove this formula, I am sure you can to it).

According to law of Archimedes, the buoyant force pushing the ball upwards is equal to the weight of the water displaced by the body, i.e., the weight of the water which would have filled the submerged part of the body. If $g = 9.8\, \frac{\text{m}}{\text{s}^2}$ stands for the free-fall acceleration, then the buoyant force is equal to

$$F_{\text{buoyant}} = g\, \rho_{\text{water}}\, V_{\text{submerged}} \ .$$

The ball will be in equilibrium if the weight of the ball,

$$F_{\text{weight of ball}} = g\, \rho_{\text{ball}}\, V_{\text{ball}} \ ,$$

(pulling the ball downwards) is equal to the buoyant force F_{buoyant} (which pushes the ball upwards):

$$g\, \rho_{\text{water}}\, V_{\text{submerged}} = g\, \rho_{\text{ball}}\, V_{\text{ball}} \ .$$

Canceling out g and using the formula for the volume of a ball, we obtain the following condition for equilibrium:

$$\rho_{\text{water}}\, \frac{\pi}{3}\, z^3\, (3r - z) = \rho_{\text{ball}}\, \frac{4\pi}{3} r^3 \ .$$

Let s be a (positive) dimensionless quantity defined as

$$s = \frac{\rho_{\text{ball}}}{\rho_{\text{water}}} > 0 \ .$$

Using this definition, after some elementary algebra one can rewrite the equilibrium condition in the form

$$z^3 - 3rz^2 + 4sr^3 = 0 \ . \tag{4}$$

Finally, let us introduce the non-dimensional quantity

$$\xi = \frac{z}{r} \ ,$$

whose physically meaningful range is, obviously, $\xi \in [0, 2]$. In terms of ξ, the equilibrium condition (4) takes the simple form

$$\xi^3 - 3\xi^2 + 4s = 0 \ . \tag{5}$$

(a) Check that the expression (3) for the volume of the submerged part is plausible. In other words, do *not* prove it, but think of several – I want you to think of three – different cases in which this formula should give obvious results (for example, it is obvious what $V_{\text{submerged}}$ should be if $z = 0$ – check that (3) indeed gives you this value for $V_{\text{submerged}}$; then think of two more obvious cases).

(b) It is clear from the physics of the problem that for s in the physically meaningful range, the equation (5) must have a unique solution ξ^* in the physically meaningful range of values of ξ. You have to show mathematically that this is indeed the case.

To do this, I suggest that you do the following. Let $f(\xi) = \xi^3 - 3\xi^2 + 4s$ stand for the left-hand side of the equilibrium condition (5). The function f is a cubic polynomial, hence it cannot have more than three real zeros, and can have no more than two extrema. Show that the function f always has a local maximum at the point $\xi_1 = 0$. What is the value ξ_2 of ξ for which f has a local minimum? What are the values of $f(\xi_1)$ and $f(\xi_2)$ if s is such that the ball floats (i.e., does not sink)? Is the function f continuous? Use all these facts to show mathematically (by using some of the theorems we have discusses in class) that if s in the "floating" range, the equation $f(\xi) = 0$ has a unique physically meaningful solution. Which theorem have you used to draw this conclusion?

(c) Use the MATLAB program `newton.m` (which you can find at the class web-site) to apply Newton's method for computing the numerical value of ξ^* for $s = 0.2$, 0.4, 0.5 (this is obvious!), 0.6, and 0.8, with tolerance 10^{-8}; start from some reasonable initial value. Write your results in a table containing the value of x, the corresponding values of ξ^*, and the number of steps that Newton's method needed to provide the necessary precision. If the value of ξ^* you obtain is not physically reasonable, change the initial point (remember, Newton's method gets lost very easily). Please attach the printout from running the program.

(d) Continue your reasoning in part (b) to show that if $s > 1$, there is only one real solution of the equilibrium condition (5), and this solution is not physically reasonable.

Problem 3. Suppose that $(p_n)_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n. If positive constants λ and α exist such that

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda \ , \tag{6}$$

then we say that the sequence $(p_n)_{n=0}^{\infty}$ *converges to p of order α, with asymptotic error constant λ.* The concepts of asymptotic error constant λ and especially order of convergence α are very important when one is using an *iterative method*, i.e., a method in which the exact solution of the problem

is found as a limit of a sequence of approximate values. If the exact value p is a limit of a sequence $\{p_n\}_{n=0}^{\infty}$ of approximate values, then the *error* at the nth step of the iteration is $E_n := |p_n - p|$. The rate of decreasing of E_n is one of the most important characteristics of an iterative method.

Assume that the sequence $(p_n)_{n=0}^{\infty}$ is generated by some iterative method for finding a root of an equation. Also assume that we know that the sequence $(p_n)_{n=0}^{\infty}$ converges to some number p of some order α with some asymptotic error constant λ, but we don't know the values of α and λ. The goal of this problem is to develop a method for determining the numerical value of α from the numerical values of the members of the sequence $(p_n)_{n=0}^{\infty}$. Let $\ell_n := \log_{10} E_n$.

(a) Show that for large n, the following approximate identity holds:

$$\ell_n - \alpha \ell_{n-1} \approx \log_{10} \lambda \ .$$

Hint: Just look at the definition (6) of order of convergence.

(b) Using the approximate identity derived in (a) show that

$$\alpha \approx \frac{\ell_n - \ell_{n+1}}{\ell_{n-1} - \ell_n} \ .$$

Note that this approximate formula for α does not depend on the base of the logarithms; if ℓ_n is defined as the log base 10 of E_n, the formula will remain the same.

(c) The data in Table 1 come from applying the *Newton method* and the *secant method* to find the root of the equation

$$x + \sin x = 1 \ ,$$

whose exact value is $p = 0.510973429388569109520013971145080632045358892 62\ldots$. Use

Table 1: Log_{10} of the errors of the Newton and the secant methods.

n	ℓ_n, Newton	ℓ_n, secant
0	−0.31067	−0.31067
1	−2.85988	−1.49389
2	−7.84087	−2.54052
3	−17.7179	−4.90935
4	−37.4715	−8.33484
5	−76.9787	−14.1282
6	−155.993	−23.3471
7	−314.022	−38.3595
8	−630.079	−62.5907
9	−1262.19	−101.834
10	−2526.42	−165.309
11	−5054.88	−268.027
12	−10111.8	−434.220

the formula derived in part (b) to find empirically the order of convergence α for these two methods.

Problem 4. On August 2, 2010, Shigeru Kondo used Alexander Yee announced that they have calculated 5,000,000,000,000 digits of π (on October 17, 2011 they improved their own record by computing 10 trillion digits of π). They used the a program called y-cruncher, developed by Yee, and performed their computations on a single desktop computer built by Kondo; the computation took 90 days (between 6:19 p.m. on May 4 and 1:21 p.m. on August 3, 2010). You can see their announcement and details on their work at

`http://www.numberworld.org/misc_runs/pi-5t/announce_en.html`

`http://www.numberworld.org/misc_runs/pi-5t/details.html`

In their computations Kondo and Yee used the following formula derived by the brothers David and Gregory Chudnovsky, who relied on some ideas of the famous Inidian mathematician Srinivasa Ramanujan (1887–1920):

$$\frac{1}{\pi} = \frac{\sqrt{10005}}{4270934400} \sum_{k=0}^{\infty} \frac{(-1)^k (6k)!}{(k!)^3\,(3k)!} \, \frac{13591409 + 545140134k}{640320^{3k}} \ .$$

In this problem you will use Mathematica to find the rate of convergence of the right-hand side of this formula to the exact value of $\frac{1}{\pi}$. You can define the function `chud[n]` which computes the sum of the first `n` terms of Chudnovsky's formula:

```
termPi[k_]=(-1)^k*(6*k)!/(k!)^3/(3*k)!*(13591409+545140134*k)/640320^(3*k)

chud[n_]=Sqrt[10005]/4270934400*Sum[termPi[k], {k, 0, n}]
```

After you type each line in Mathematica, press SHIFT, hold it down, and press RETURN. The underscores after `k` and `n` in `termPi[k_]` and `chud[n_]` tell Mathematica that we are defining new functions, and `k` and `n` the variables of these functions.

To find the numerical value with accuracy of 1000 digits of the difference between the exact value of $\dfrac{1}{\pi}$ and the partial sum of the sum containing, say, 8 terms – which in our notations will be equal to `chud[7]` – you can type the following:

```
N[chud[7] - 1/Pi, 1000]
```

There will a problem, however, and Mathematica will complain that its internal precision limit is not enough for the computation (try it!). That is why you have to type

```
Block[{$MaxExtraPrecision = 1000}, N[chud[7] - 1/Pi, 1000]]
```

(a) Compute the numerical values of the absolute error $E_n = \left|\frac{1}{\pi} - \texttt{chud}[n]\right|$ for $n = 0, 1, 2, 3, 4, 5, 6, 7$, and write your results in a table (there is no need to write more than 3–4 digits of accuracy of E_n in the table).

(b) For the values of n used in part (a), show that your numerical results give $\frac{E_{n+1}}{E_n} \approx 10^{-14}$. Can you express E_n approximately in terms of E_0? I do not want anything sophisticated, just a VERY ROUGH approximate formula.