May 3, 2021

Numerical methods and mathematical modelling
Mid-term projects

doc. dr. sc. Kosuke Nomura[1]

1 Numerical accuracies

Let us consider the Taylor expansion of the function $\sin(x)$:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} \tag{1}$$

In the mathematical sense, the series converges for all values of x. Accordingly, a reasonable algorithm to compute the function $\sin(x)$ could be to consider the finite sum:

$$\sin x \simeq \sum_{n=1}^{N} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} \tag{2}$$

which means the Taylor series stops at the N-th term.

Problem.

1. Write a program in C/C++ for calculating $\sin(x)$ as the finite sum defined above;

2. Calculate the series for $x \leqslant 1$ and compare it with some built-in function like in Python (which is assumed to be exact). Stop the summation at N for which the next term in the series will be no more than 10^{-7} of the sum up to that point, b.z.,

$$\frac{|(-1)^N x^{2N+1}|}{(2N-1)!} \leqslant 10^{-7} \left| \sum_{n=1}^{N} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} \right| \tag{3}$$

3. Examine the terms in the series for $x \simeq 3\pi$, and observe the significant subtractive cancellations that occur when large terms add together to give small answers. Print-out near-perfect cancellation around $n \simeq x/2$;

4. Examine if better precision is obtained by using trigonometric identities to keep $0 \leqslant x \leqslant \pi$;

5. By progressively increasing x from 1 to 10, and then from 10 to 100, use your program to determine experimentally when the series starts to lose accuracy and when it no longer converges;

6. Make a series of plots of the error v.s. N for different values of x.

[1] E-mail: knomura@phy.hr

2 Numerical Integration

Consider radioactive decay of atomic nuclei. Let $N(t)$ denote the number of nuclei at time t, which decay as time goes. An experiment has measured $\frac{dN(t)}{dt}$, the number of particles entering a counter per unit time.

Problem. Consider to integrate this spectrum to obtain the number of particles $N(t)$ that entered the counter in the first second.

$$N(1) = \int_0^1 \frac{dN(t)}{dt} dt \tag{4}$$

1. Write a C/C++ program to integrate a function numerically using the Trapezoid rule, as well as the Simpson rule. Use double precision. It is also encouraged to try to implement in your progmra the Gaussian quadrature (cf. Chapt. 4.4 of Morten Hjorth-Jensen's textbook). Suppose that the particles decaying exponentially

$$\frac{dN(t)}{dt} = e^{-t} \Longrightarrow N(1) = \int_0^1 e^{-t} dt = 1 - e^{-1} \tag{5}$$

2. Compute the relative error, $\epsilon = |(\text{numerical} - \text{exact})/\text{exact}|$, in each case of the algorithms. Present you results in a table/plot. Try different numbers of integration poits N, e.g., of 2, 10, 20, 40, 80, 160, ...

3. Make a log-log plot of relative error v.s. N (the number of integration points).

4. Observe that the error is of a power-law dependence on the number of points N, i.e.,

$$\epsilon \simeq CN^\alpha \Longrightarrow \alpha \log N + const. \tag{6}$$

   Using your plot/table, estimate this power-law dependence in the cases of Trapezoidal and Simpson's rules and for the regimes where the algorithmic and round-off errors appear. Determine the number of decimal places of precision in your calculation [2].

3 Ordinary Differential Equation

Consider again a one-dimensional, classical object with mass m attached to a spring that exerts a restoring force toward the origin. There is also a time dependent external force on the mass. The equation of motion reads

$$F_k(x) + F_{ext}(x,t) = m\frac{d^2x}{dt^2} \tag{7}$$

where $F_k(x)$ is the restoring force exerted by the spring and $F_{ext}(x,t)$ is the external force. Here we assume that the spring's potential function is proportional to some arbitrary *even* power p of the displacement x from equilibrium:

$$V(x) = \frac{1}{p} k x^p. \tag{8}$$

[2] A power-law dependence appears as a straight line on a log–log plot, and if you use $\log_{10}$, then the ordinate on your log–log plot will be the negative of the number of decimal places of precision in your calculation

We require an even p to ensure that the force,

$$F_k(x) = -\frac{dV(x)}{dx} = -kx^{p-1}, \tag{9}$$

contains an odd power of p, which guarantees that it is a restoring force for positive and negative x values. We need to solve the second-order ODE:

$$m\frac{d^2x}{dt^2} = F_{ext}(x,t) - kx^{p-1}. \tag{10}$$

In what follows, we omit the external force $F_{ext}(x,t)$ to simplify the problem.

Problem. Use your fourth-order Runge-Kutta `rk4` program in C/C++ to study anharmonic oscillations by trying powers in the range $p = 2 - 12$. Note that for large values of p, the forces and accelerations get large near the turning points, and so you may need a smaller step size h than that used for the harmonic oscillator. Try to answer the following points.

1. Check that the solution remains periodic with constant amplitude and period for a given initial condition regardless of how nonlinear you make the force. In particular, check that the maximum speed occurs at $x = 0$ and that the velocity $v = 0$ at maximum x's, the latter being a consequence of energy conservation.

2. Verify that nonharmonic oscillators are *nonisochronous*, that is, vibrations with different amplitudes have different periods.

3. Explain why the shapes of the oscillations change for different powers p.

4. Devise an algorithm to determine the period T of the oscillation by recording times at which the mass passes through the origin. Because the motion can be asymmetric, you must record at least three times to deduce the period.

5. Make a graph of the deduced period as a function of initial amplitude.

6. Verify that the motion is oscillatory, but not harmonic, from certain value of p on.