NATIONAL UNIVERSITY OF SINGAPORE

PC5215 – NUMERICAL RECIPES WITH APPLICATIONS

(Semester I: AY 2019-20)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. Please write your student number only. Do not write your name.
2. This assessment paper contains FOUR questions and comprises THREE printed pages.
3. Students are required to answer ALL questions; questions carry equal marks.
4. Students should write the answers for each question on a new page.
5. This is a CLOSED BOOK examination.
6. Non-programmable calculators are allowed.

1. Consider a two-point Gaussian quadrature formula in the domain [0,+∞) with an exponential decaying weight:

$$\int_0^{+\infty} f(x)e^{-x}\,dx \approx w_1 f(x_1) + w_2 f(x_2).$$

   a. Determine the first three orthogonal polynomials $P_0(x)$, $P_1(x)$, $P_2(x)$, necessary to fix the abscissas x_1 and x_2.
   b. Determine the abscissas x_1 and x_2 and the weights w_1 and w_2.
   c. Quantify the error of this Gaussian quadrature formula.

   a. For simplicity, we can take the coefficient of the highest power to be 1, so $P_0 = 1$, $P_1 = x + c$. We determine c by the orthogonality requirement, $\langle P_0|P_1\rangle = 0$, or $\int_0^{+\infty}(x+c)e^{-x}dx = 0$. We can perform the integral using a general formula: $\int_0^{\infty} x^n e^{-x}dx = n!$ This gives 1 + c = 0, or c = -1. P_1=x-1. For the quadratic polynomial, we assume $P_2 = x^2 + bx + d$. P_2 must be orthogonal to P_0 and P_1. This determine the two coefficients b and d to be -4 and 2. So $P_2 = x^2-4x+2$.

   b. x_1 and x_2 are the roots of P_2, $x^2-4x+2 = 0$, given $x_1 = 2-\sqrt{2}$, $x_2=2+\sqrt{2}$. Take the general f(x) to be 1 and x, we got two equations that can determine w_1 and w_2. They are $w_1+w_2 = 1$, $w_1(2-\sqrt{2}) + w_2(2+\sqrt{2}) = 1$. This gives $w_1 = (2+\sqrt{2})/4$, $w_2=(2-\sqrt{2})/4$.

   c. The two-point Gaussian quadrature formula is exact for all polynomials of degree 3 or less. Error occurs at x^4.

2. Consider Monte Carlo sampling of a one-dimensional Ising model with energy function

$$H(\sigma) = -J\sum_{i=1}^{N}\sigma_i\sigma_{i+1}$$

in the canonical ensemble. We assume periodic boundary condition, that is, $\sigma_{N+i} = \sigma_i$. N is some arbitrary natural number larger than 1, and the coupling parameter $J > 0$, $\sigma_i = \pm 1$. The Gibbs sampling is slightly different from the Metropolis algorithm, as follows: first, pick a site $1 \le i \le N$ at random. Compute $= \frac{x}{x+x^{-1}}$, here $x = e^{\beta J(\sigma_{i-1}+\sigma_{i+1})}$, $\beta = 1/(k_B T)$. Then set σ_i as +1 with probability p, or -1 with probability $1-p$.

   a. Write down the associated transition matrix $W(\sigma \to \sigma')$.
   b. Show that the detailed balance condition is satisfied, that is,

$$P(\sigma)W(\sigma \to \sigma') = P(\sigma')W(\sigma' \to \sigma).$$

      Here $P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z}$, and Z is the partition function.

   c. Give a complete Python code to realize this algorithm.

For N sites, the transition matrix W is 2^N by 2^N. We can write W as $T(\sigma \to \sigma')P(\sigma'|\sigma)$. Here T is 1/N if the new state σ' differs from the old state σ by 1 spin (say at site i), and 0 otherwise. The last factor is the conditional probability where i-th site involved. I.e., $P(\sigma'|\sigma) = P(\dots\sigma_i'\dots)/[P(\dots,\sigma_i=+1,\dots)+P(\dots,\sigma_i=-1,\dots)]$. Take the ratio, we can see detailed balance is satisfied. Python code omitted.

3. To minimize a function $F(\boldsymbol{x})$ in high dimensions in the contexts of neural networks, the simplest method is the "stochastic" gradient descent, according to $\boldsymbol{x} \to \boldsymbol{x} - \eta \nabla F(\boldsymbol{x})$. Here $\boldsymbol{x}$ is a vector and ∇ denotes gradient with respect to $\boldsymbol{x}$.
    a. Explain the meaning of the term "stochastic" as used in learning algorithms of neutral networks.
    b. Consider a simple function $F(\boldsymbol{x}) = \frac{1}{2}x_1^2 + x_2^2 + x_1 x_2$, defined in two-dimensional space $\boldsymbol{x} = (x_1, x_2)$. If we start from the position $\boldsymbol{x}_0 = (1,1)$ with $\eta = 0.1$, determine the new position after two steps of the descent algorithm.
    c. The conjugate gradient (CG) method is a much faster algorithm. Do the same as in part b, starting again from $\boldsymbol{x_0} = (1,1)$, apply CG method in two steps.

a. Stochastic (gradient descent) in the context of learning algorithms means we do not use the complete data set to evaluate F. Instead, we choose a smaller set, "at random", so that F can be computed faster.

b. The gradient vector is $\nabla F = (\partial F/\partial x_1, \partial F/\partial x_2) = (x_1 + x_2, x_1 + 2x_2)$. So at step 0, gradient is (2,3). This gives x_1 =(1,1)-0.1(2,3) = (0.8, 0.7). At x_1 the gradient becomes (1.5, 2.2). So x_2 = (0.8,0.7) – 0.1(1.5,2.2) = (0.65, 0.48). This is still far away from the exact answer, which is (0,0).

c. The first step in CG is the same as in b except we minimize F with respect to η. The gradient is same as in b, so we have $x_1 = 1 - 2\eta, x_2 = 1 - 3\eta$. $\partial F/\partial \eta = 0$ gives $\eta = 13/34 \approx 0.382$. So at this point, x_1 = (0.235, -0.147). For the second step, we compute $\gamma = |$new gradient / old gradient$|^2$ = 0.000865. Now the new search direction is $n = -\nabla F + \gamma n(old)$=(-0.08996,0.05622). Perform again a 1D line search starting from x_1 and direction n, that is $x_2 = x_1 + \lambda n$, we find optimal value $\lambda = 2.61538$. This gives final answer x_2=(0,0), which is the location of minimum of F , i.e., CG method terminates in two steps. No more search is needed.

4. The harmonic oscillator has the classical Hamiltonian

$$H(x,p) = \frac{p^2}{2m} + \frac{1}{2}Kx^2.$$

Here p is momentum, x is position, m is mass, and K is force constant. The system is described by the Hamilton equations of motion.
    a. Give the Euler algorithm with time step h for a numerical solution (x_n, p_n), $n = 0,1,2, \ldots$, of the Hamilton equations of the oscillator. Also indicate the local truncation error.
    b. The total energy of the oscillator at the n-th step is $E_n = \frac{p_n^2}{2m} + \frac{1}{2}Kx_n^2$. Discuss what happens to the total energy in the limit $n \to \infty$. (Hint: analyze the eigenvalues of the 2 by 2 matrix that transforms from the old state to the new state).

c. Revise your algorithm in part a to a symplectic algorithm. And discuss again the behavior of the total energy as the number of steps n goes to infinity. In answering question b and c, we disregard the round-off error due to finite machine precision.

a. Using the Euler method, we can compute the momentum and position at n+1 step from n step according to the matrix equation $\begin{pmatrix} p_{n+1} \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & -Kh \\ \frac{h}{m} & 1 \end{pmatrix} \begin{pmatrix} p_n \\ x_n \end{pmatrix}$. *The local truncation error is O(h²).*

b. The eigenvalue of the matrix is determined from $det \begin{pmatrix} \lambda - 1 & Kh \\ -\frac{h}{m} & \lambda - 1 \end{pmatrix} = 0$. *The eigenvalues are* $\lambda = 1 \pm ih\sqrt{K/m}$. *Then* $|\lambda| = 1 + \frac{Kh^2}{m} > 1$. *Since the eigenvalues of the matrix is larger than 1, after repeated multiplications, the magnitude of p and x will increase, and the energy diverges to infinite as n goes to infinity. Note after n-th iterations, the result is the matrix to the n-th power, which we diagonalize and result in the eigenvalues to the n-th power.*

c. For a symplectic algorithm, we can make a move in p and then followed by x using the new value of p, the matrix can be written as $\begin{pmatrix} p_{n+1} \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & -Kh \\ \frac{h}{m} & 1 - Kh^2/m \end{pmatrix} \begin{pmatrix} p_n \\ x_n \end{pmatrix}$. *The eigenvalues can now be determined in the same way to be* $\lambda = 1 - \frac{u^2}{2} \pm iu\sqrt{1 - \frac{u^2}{4}}$, *here* $u = h\sqrt{\frac{K}{m}}$. *We can check that the magnitude of the eigenvalues is exactly 1,* $|\lambda| = 1$. *As a result, the iteration does not divergence and the energy is approximately conserved. This shows the advantage of symplectic method over the Euler method.*

--- the end --- [WJS]