Computational Physics Final Examin

Avijit Maity DNAP (Int. PhD)

1) The system of equation is given by —

$$\begin{bmatrix} 4 & 1 & 2 \\ 2 & 4 & -1 \\ 1 & 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ -5 \\ -9 \end{bmatrix}$$

In this case the augmented matrix is $\left[\begin{array}{ccc|c} 0.4\times10^1 & 0.1\times10^1 & 0.2\times10^1 & 0.9\times10^1 \\ 0.2\times10^1 & 0.4\times10^1 & -0.1\times10^1 & -0.5\times10^1 \\ 0.1\times10^1 & 0.1\times10^1 & -0.3\times10^1 & -0.9\times10^1 \end{array}\right]$

As the calculation is being done on a decimal computer capable of carrying only two floating point digits, we will apply most general rule.

The method proceeds along following steps

I) $R_1 \rightarrow R_1 / 0.4\times10^1$

$$\left[\begin{array}{ccc|c} 0.1\times10^1 & 0.25 & 0.50 & 0.23\times10^1 \\ 0.2\times10^1 & 0.4\times10^1 & -0.1\times10^1 & -0.5\times10^1 \\ 0.1\times10^1 & 0.1\times10^1 & -0.3\times10^1 & -0.9\times10^1 \end{array}\right]$$

II) $R_2 \rightarrow R_2 - 0.2\times10^1 \times R_1$, $R_3 \rightarrow R_3 - R_1$

$$\left[\begin{array}{ccc|c} 0.1\times10^1 & 0.25 & 0.50 & 0.23\times10^1 \\ 0 & 0.35\times10^1 & -0.2\times10^1 & -0.96\times10^1 \\ 0 & 0.75 & -0.35\times10^1 & -0.11\times10^2 \end{array}\right]$$

III) $R_2 \rightarrow R_2 / (0.35\times10^1)$

$$\left[\begin{array}{ccc|c} 0.1\times10^1 & 0.25 & 0.50 & -0.23\times10^1 \\ 0 & 0.1\times10^1 & -0.57 & -0.27\times10^1 \\ 0 & 0.75 & -0.35\times10^1 & -0.11\times10^2 \end{array}\right]$$

IV) $R_3 \rightarrow R_3 - R_2 \times 0.75$

$$\left[\begin{array}{ccc|c} 0.1\times10^1 & 0.25 & 0.50 & -0.23\times10^1 \\ 0 & 0.1\times10^1 & -0.57 & -0.27\times10^1 \\ 0 & 0 & -0.31\times10^1 & -9 \end{array}\right]$$

Scanned with CamScanner

Thus by back substitution —

$$x_3 = -\frac{0.78\times10^1}{0.35\times10^1} = -0.22\times10^1$$

$x_2 = 0.27\times10^1$

$x_1 = 0.23\times10^1 - x_2\times0.25 - x_3\times0.50$

$=$

v) $R_3 \to R_3/(-0.31\times10^1)$

$$\left[\begin{array}{ccc|c} 0.1\times10^1 & 0.25 & 0.50 & 0.23\times10^1 \\ 0 & 0.1\times10^1 & -0.57 & -0.27\times10^1 \\ 0 & 0 & 0.1\times10^1 & +0.29\times10^1 \end{array}\right]$$

By back substitution —

$x_3 = +0.29\times10^1$

$x_2 = -0.27\times10^1 + 0.57\times x_3 = -0.1\times10^1$

$x_1 = -0.23\times10^1 - 0.25\times x_2 - 0.50\times x_3$

$= 0.11\times10^1$

So, the solution is —

$x = \{0.11\times10^1, -0.1\times10^1, 0.29\times10^1\}$

I have calculated the solution using nump.linalg.solve and we get $= (1,-1,3)$ which is almost same.

Problem-2

a) Fourier transform of a sample :-

Python :- for finding FFT : np.fft.fft()
for frequency : np.fft.fftfreq()

C :-

fftw_plan_r2r_1d()
fftw_plan_r2r_2d() } in fftw

b) QR decomposition of a matrix :-

Python :- numpy.linalg.qr()
scipy.linalg.qr()

c) A million random numbers from a lognormal PDF :-

Python :- numpy.random.Generator.lognormal.

C :- log_normal_truncated_ab();

d) 8th order runge kutta method :-

Python :- scipy.integrate.solve_ivp (method = "DOP 853")

C :- gsl_odeiv2_step_rk8pd
with header file gsl_odeiv2.h

e) singular value decomposition :-

Python :- numpy.linalg.svd()

f) Sampling a 5048 dimensional PDF :-

Python :- scipy.stats()
numpy.random

g) Adaptive step size control :-

Python :- 1) scipy.integrate.LSODA (fun, t0, y0, t_bound, first_step = None, min_step = 0.0, max_step = inf, rtol = 0.001, atol = 1e-06, Jac = None, lband = None, uband = None, vectorized false

ii) scipy. integrate. solve_ivp (func, [t_min, t_max], Y_initial, method= 'LSODA')]

C :- gsl_odeiv2_control
with header file gsl_odeiv2.h

h) 3 dimensional function using monte carlo :-

Python :- mcint. integrate (integrand, sampler(), measure, n)

C :- gsl_monte_function () ;

i) Solve a boundary value problem for 3 coupled ODE :-

Python :- scipy. integrate. odeint ()

C :- gsl_odeiv2_system

j) 10X10 complex matrix :-

Python :- numpy. linalg. eig ()
scipy. linalg. eig ()

$\boxed{\text{Problem-3}}$

A tridiagonal system for n unknowns may be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

where $a_1 = 0$, $c_n = 0$

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & c_{n-2} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_n \end{bmatrix}$$

So, augmented matrix is

$$= \left[\begin{array}{ccccc|c} b_1 & c_1 & & & 0 & d_1 \\ a_2 & b_2 & c_2 & & & d_2 \\ & a_3 & b_3 & \ddots & c_{n-1} & \vdots \\ & & \ddots & a_n & b_n & d_n \end{array}\right]$$

$$= \left[\begin{array}{ccccc|c} a_{11} & a_{12} & & & 0 & d_1 \\ a_{21} & a_{22} & a_{23} & & & d_2 \\ & a_{32} & a_{33} & & & d_3 \\ & & \ddots & \ddots & & \vdots \\ & & & a_{nn-1} & a_{nn} & d_n \end{array}\right]_{n \times (n+1)}$$

Step-1 (Triangularization)

~~Step-1~~ $R_1 \to R_1 / a_{11}$, $R_2 \to R_2 - a_{21} \times R_1$ which will yield

$$\left[\begin{array}{ccccc|c} 1 & a'_{12} & & & 0 & d'_1 \\ 0 & a'_{22} & a_{23} & & & d'_2 \\ & & \ddots & & & \vdots \\ & & & a_{nn-1} & a_{nn} & d_n \end{array}\right]_{n \times (n+1)}$$

where, $a'_{12} = \dfrac{a_{12}}{a_{11}}$

$a'_{22} = a_{22} - \dfrac{a_{21} a_{12}}{a_{11}}$

For this step, $i=1$ we need 3 divisions, 3 multiplication and 3 subtraction (because the element above 23 is zero so no computation needed). So total step

$= 3+3+3 = 9$

Now, next step $\Rightarrow R_2 \to \frac{R_2}{a_{22}'}$, $R_3 \to R_3 - a_{32}\cdot R_2$

we will get

$$\left[\begin{array}{cccccc|c} 1 & a_{12}' & \cdots & & & 0 & d_1' \\ 0 & 1 & a_{23}' & & & & d_2'' \\ 0 & 0 & a_{33}' & a_{34} & & & d_3' \\ & & & \ddots & & & \vdots \\ & & & & & & d_n \end{array}\right]$$

For this step, we will have total step $= 3+3+3 = 9$

so, we only need 9 computation for $(n-1)$ steps & 2 computation for the last step which is just ~~three~~ two divisions to make the last diagonal term unity.

so, forword sweep with normalization—

$$\gamma_1 = c_1/b_1, \quad \gamma_K = \frac{c_K}{b_K - a_K\gamma_{K-1}} \quad \text{for } K = 2,3 \cdots (n-1)$$

$$\beta_1 = \frac{d_1}{b_1}, \quad \beta_K = \frac{d_K - a_K\beta_{K-1}}{b_K - a_K\gamma_{K-1}}, \text{ for } K = 2,3 \cdots (n-1)$$

This sequence of operations finaly results in the following system of equation:—

$$\left[\begin{array}{cccccc|c} 1 & \gamma_1 & 0 & \cdots & & 0 & \beta_1 \\ 0 & 1 & \gamma_2 & \cdots & & 0 & \beta_2 \\ \cdots & 0 & 1 & \cdots & & & \vdots \\ \cdots & & & & & \gamma_{n-1} & \vdots \\ 0 & 0 & \cdots & & & 1 & \beta_n \end{array}\right]_{n\times(n+1)}$$

so, total steps $= 9(n-1) + 2 = 9n - 7$

Step-2 (backword sweep)

This leads to solution vector

$$x_n = \beta_n$$

$$x_K = \beta_K - \gamma_K x_{K+1} \quad \text{for } K = (n-1), (n-2), \cdots 1$$

which is two computation per step.

so, total computation for this backward sweep

$= 2(n-1)$

Thus, total no. of steps for tridiagonal matrix

$$= \underbrace{(9n-7)}_{\text{forward}} + \underbrace{2(n-1)}_{\text{backward}} = 11n - 9 \sim O(n)$$

—(Proved)

Problem 4(a)

From wiener $\overset{\text{Khinchin}}{\wedge}$ theorem —

$$P(K) = \int_{-\infty}^{+\infty} d\epsilon \left[\lim_{X\to\infty} \frac{1}{2X} \int_{-X}^{X} dx_1\, R(x_1, x_1+\epsilon)\right] e^{-iK\epsilon}$$

where $R(x_1, x_1+\epsilon) = E\left[f_X(x_1) f_X(x_1+\epsilon) \right]$ ~~where R~~

$$f_X(x) = \begin{cases} f(x), & x \leq X \\ 0, & x > X \end{cases}$$

For our case, $f(x) = x$

So, $R(\epsilon) = \lim_{X\to\infty} \frac{1}{2X} \int_{-X}^{+X} dx_1\, R(x_1, x_1+\epsilon)$

So, $P(K) = \int_{-\infty}^{+\infty} R(\epsilon)\, e^{-iK\epsilon}\, d\epsilon$ | ~~R(ϵ) is com~~ for uniform distribution $R(\epsilon) = 1$

$= 2\pi\delta(K)$

So, power spectral density is delta function centered around $K=0$

—(Justified)

Problem-5

Criteria to choose libraries —

1) The first criteria to choose library function is the speed. The faster the function in library, the better

2) The function in library generate various intermediate arrays and variable which need storage. so, storage is required for such arrays and variables.

3) Accuracy of the result produced using the library is also an important criteria.

4) The next criteria is user-friendly. The library must be compatible with the language in which I do most of my coding work.

Scanned with CamScanner

Problem-6

$$y_1' = 32y_1 + 66y_2 + 2/3\,x + 2/3 \quad \text{--- (1)}$$

$$y_2' = -66y_1 - 133y_2 - 1/3\,x - 1/3 \quad \text{--- (2)}$$

$2\times(1) + (2)$, we get —

$$2y_1' + y_2' = -2y_1 - y_2 + (x+1)$$

$$\Rightarrow (2y_1+y_2)' + (2y_1+y_2) = (x+1)$$

or, $f' + f = (x+1)$ where $f = 2y_1 + y_2$

$$f(0) = 2y_1(0) + y_2(0) = 2/3 + 1/3 = 1$$

so, its solution is —

$$f(x) = e^{-x} + x$$

If I plot the graph of $2y_1+y_2$, $f(x)$ then we can show it is perfectly similar to each other. So, the solution obtained is justified. I have plotted this in python.

Problem-7 :

A linear conguential psedorandom generators has four parameter : modulus (m), multiplier (a), increment (c), seed (x_0).

Then the sequence of random numbers is obtained by $x_{i+1} = (ax_i + c) \bmod m$

* The linear Congeuential Generator completely breaks down if the number m, a, and c are not choosen carefully, For example, if I take $m=10$, $a=8$, $c=8$, then with initial seed $x_0 = 8$, I get the repeating sequence 2, 4, 0, 8, 2, 4, 0, 8 i.e initial seed comes back again.

* we can give an example of generator in which the seed never appears again in the sequence.

If, we choose $a = 4$, $c = 0$, $m = 2^4 = 16$, $x_0 = 1$

Then we get, $x_1 = (ax_0 + c) \bmod m$

$= (4 \times 4 + 0) \bmod 16 = 0$

$x_2 = (ax_1 + c) \bmod m = (a \times 0 + 0) \bmod 16 = 0$

- - - -

Then we get a sequence —

$\{4, 0, 0, 0 \cdots\}$

— here we never get 4 (initial seed) again.

also for $a = 1664525$, $c = 1013904223$

$m = 4294967296$, $x_0 = 1$

we get —

1015568748, 958600546 7, - - -