Final Exam

1. [5%] Derive the error of Trapezoidal Rule.
2. [10%] Derive the Adams-Bashforth Four-Step explicit method.
3. [5%] Using Secant method to determine the highest real root of the following equation (three iterations, $x_{-1} = 3, x_0 = 4$).

$$f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$$

4. [5%] Use both Gauss elimination and LU decomposition to decompose the following system. Please show all the steps in the computation.

$$x_1 + 7x_2 - 4x_3 = -51$$
$$4x_1 - 4x_2 + 9x_3 = 62$$
$$12x_1 + x_2 + 3x_3 = 8$$

5. [10%] Develop cubic splines for the following data points and predict f(4) and f(2.5)

X	1	2	3	5	7	8
f(x)	3	6	19	99	291	444

6. [10%] Use 3-point and 4-point Gaussian methods to integrate $\int_0^2 e^{-\cos^2 x}dx$ and $\int_0^1 dx/(e^x\sqrt{x})$

Number of points, n	Points, x_i	Approximately, x_i	Weights, w_i	Approximately, w_i
1	0	0	2	2
2	$\pm\frac{1}{\sqrt{3}}$	±0.57735	1	1
3	0	0	$\frac{8}{9}$	0.888889
	$\pm\sqrt{\frac{3}{5}}$	±0.774597	$\frac{5}{9}$	0.555556
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.339981	$\frac{18+\sqrt{30}}{36}$	0.652145
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.861136	$\frac{18-\sqrt{30}}{36}$	0.347855

7. [15%] Consider the third-order Runge-Kutta method:

$$x(t+h) = x(t) + \frac{1}{9}(2K_1 + 3K_2 + 4K_3)$$

where

$$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf\left(t + \frac{1}{2}h, x + \frac{1}{2}K_1\right) \\ K_3 = hf\left(t + \frac{3}{4}h, x + \frac{3}{4}K_2\right) \end{cases}$$

a. Show that it agrees with the Taylor series method of the same order for the differential equation $x' = x + t$. [6%]

b. Prove that this third-order Runge-Kutta method reproduces the Taylor series of the solution up to and including terms in h^3 for any differential equation. [9%]

Computer questions

8. [15%] Solve the following initial-value problem :

$$y' = te^{3t} - 2y \ , for \ 0 \le t \le 1 \ , with \ y(0) = 0 \ and \ h = 0.1$$

$$actual \ solution \ y(t) = \frac{1}{5}te^{3t} - \frac{1}{25}e^{3t} + \frac{1}{25}e^{-2t}.$$

(a) Heun's method

(b) Modified Euler method

(c) Runge-Kutta of order 4. (Runge-Kutta-Gill)

(d) Adams-Moulton closed formula. (N=3.4.5)

and compare the results to the actual values.

9. [15%] Use the TDMA method to solve the given equation :

$$f = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \ ; \ (a) f = 3 \ , \ (b) f = 3 - \frac{1}{10}T \ .$$

$L_x = 10m, L_y = 10m$, N = 21

Boundary conditions : $\left.\frac{\partial T}{\partial x}\right|_{x=0} = 0 \ , \ T_{y=0} = 0 \ , \ T_{x=Lx} = 0 \ , \ T_{y=Ly} = 0$

10. [10%] Use the Monte Carlo method to estimate the volume of the solid whose points (x, y, z) satisfy

$$0 \le x \le y, 1 \le y \le 2, -1 \le z \le 3$$

$$\begin{cases} e^x \le y \\ (\sin z) y \ge 0 \end{cases}$$

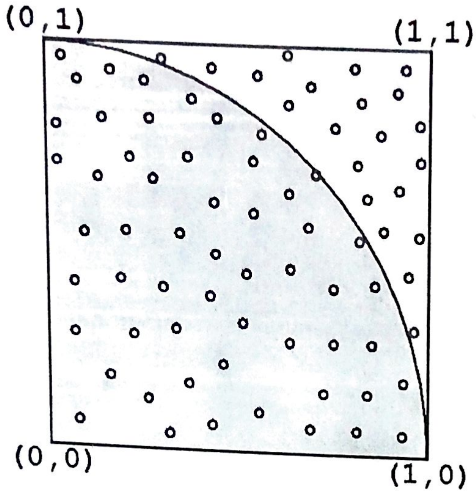
<u>Runge-Kutta-Gill Method</u>

$$x(t+h) = x(t) + \frac{1}{6}\left(K_1 + 2\left(1 - \frac{1}{\sqrt{2}}\right)K_2 + 2\left(1 + \frac{1}{\sqrt{2}}\right)K_3 + K_4\right)$$

where

$$\begin{cases} K_1 = hf(t, x) \\ K_2 = hf\left(t + \frac{1}{2}h, x + \frac{1}{2}K_1\right) \\ K_3 = hf\left(t + \frac{1}{2}h, x + \left(-\frac{1}{2} + \frac{1}{\sqrt{2}}\right)K_1 + \left(1 - \frac{1}{\sqrt{2}}\right)K_2\right) \\ K_4 = hf\left(t + h, x - \frac{1}{\sqrt{2}}K_2 + \left(1 + \frac{1}{\sqrt{2}}\right)K_3\right) \end{cases}$$

蒙地卡羅法(Monte Carlo Method)求圓周率的原理示意圖如下。正方形邊長為 1 單位長，面積為 1 平方單位；黃色扇形面積等於半徑為 1 單位長的 1/4 圓，面積為 pi/4。在正方形內均勻隨機丟石頭，落在扇型內的機率 = 扇型面積÷正方形面積=pi/4。所以只要隨機產生 N 個座標(x,y)，看看座標(x,y)落在扇形中($x^2+y^2<1$)的次數有幾次。落在扇形中的次數除以 N 再乘上 4 的數值理論上就會接近圓周率 PI。



程式中我們要不斷產生 0<=x,y<1 的座標點，使用 frand()的方式來產生即可。完整程式碼如下。

C CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

double throwPI(int N){
    int i, count;
    double x, y;

    for( count = 0, i = 0; i < N; ++i ){
        x = rand()/((double)RAND_MAX+1);
        y = rand()/((double)RAND_MAX+1);
        if( x*x + y*y < 1 ) ++count;
    }
    return 4.0 * count / N;
}

int main(void){
    int i;

    srand( time(NULL) );
    for( i = 10; i <= 10000000; i *= 10 )
        printf("%10d : %10.6lf\n", i, throwPI(i) );
}
```

下面是執行結果，理論上每次執行都會有點不同，但趨勢應該是相同的，也就是N愈大，得到的結果越接近PI。從數學上可以推估答案的收斂誤差為1/(2*Sqrt(N))。

CMD

```
       10 :   3.600000
      100 :   3.240000
     1000 :   3.176000
    10000 :   3.131200
   100000 :   3.138960
  1000000 :   3.140680
 10000000 :   3.141832
100000000 :   3.141683
```