Name and section: ______________________________

ID number: ______________________________

E-mail: ______________________________

1. (30 points) Consider the Poisson problem

$$\nabla^2 u(x, y) = f(x, y), \quad (x, y) \in \Omega = (0, 1)^2, \tag{1}$$

where

$$f(x, y) = 2x\,(y - 1)\,(y - 2x + xy + 2)\exp{(x - y)}. \tag{2}$$

Furthermore, consider **Dirichlet** boundary conditions at $\partial\Omega$. The **exact solution** of this problem is given by

$$u^{\text{exact}}(x, y) = \exp{(x - y)}x\,(1 - x)\,y\,(1 - y)\,. \tag{3}$$

(a) (5 points) Show that Eq. (3) **is** indeed the exact solution of Eq. (1).

(b) (25 points) Modify the MATLAB script `dr_chapter_3_poisson.m` utilizing the 5-point Laplacian that was discussed in class in order to solve Eq. (1) on an equidistant 2D grid ($\Delta x = \Delta y \equiv h$) with $m = 101$ interior points. Recall that a **direct method** is used to solve the underlying linear system. Also, note that you must use the **exact** solution (3) in order to specify the boundary conditions. Then make a contour and surface plots of the approximate solution and calculate the **maximum absolute error**:

$$\text{error} = \max_{1 \le i,j \le m} |u^{\text{exact}}(x_i, y_j) - U_{ij}|, \tag{4}$$

where U_{ij} stands for the approximate solution and x_i as well as y_j represent the grid points.

Attach your MATLAB script, output and figures.

2. (35 points) Consider again the Poisson problem of Question 1 and your MATLAB script implemented therein. Test your script by performing a **grid refinement** study to verify that that the finite difference method employed therein is **second-order** accurate. To do so, keep increasing the number of points m as $m = 2^N$, where, e.g., $N = 3, 4, \ldots, 10$. Note that for each m the lattice spacing is modified via $h = (b - a)/(m + 1)$ again. Present your results in a table with the following format:

- column 1: h (step-size)
- column 2: error $= \max_{1 \le i,j \le m} |u^{\text{exact}}(x_i, y_j) - U_{ij}|$ (maximum absolute error).

Finally, make a plot of the maximum absolute error against h in a log-log scale.

Attach your MATLAB script, output and figure.

3. (35 points) Modify your MATLAB scripts in Questions 1 and 2 to use the 9-point Laplacian

$$\begin{aligned}\nabla_9^2 u_{ij} &= \frac{1}{6h^2}\Big\{4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} \\ &+ u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij}\Big\}, \end{aligned} \tag{6}$$

instead of the 5-point Laplacian, and to solve the same Poisson problem as the one in Question 1 using $m = 101$ interior points and Dirichlet boundary conditions. Make a contour and surface plots in this case (you **must** obtain the same results as you did in Question 1). Subsequently, perform a **grid refinement** study as you did in Question 2 to verify that that the finite difference method employed therein is **fourth-order** accurate. Present your results in a similar way as in Question 2.

Attach your MATLAB scripts, output and figures.

Some important **remarks/hints**:

- Note that the Poisson problem at the discrete level is given by

$$\nabla_9^2 u_{ij} = f_{ij},$$

where

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12}\nabla^2 f(x,y)|_{(x,y)=(x_i,y_j)}. \tag{7}$$

Since $f(x,y)$ is given by Eq. (2), you have to calculate the Laplacian of the function $f(x,y)$ **analytically** and plug it into Eq. (7). The reason that we do not use f_{ij} as is but instead Eq. (7) is that we want to achieve a fourth-order accurate method (from theory, there is a nice cancellation of the $\mathcal{O}(h^2)$ term in the local truncation error).

- In class, we discussed about the matrix representation of the 5-point Laplacian and how is implemented in MATLAB using sparse matrices. On equally footing, the matrix representation of the 9-point Laplacian follows:

$$A = \frac{1}{6h^2}\begin{bmatrix} T & C & & & \\ C & T & C & & \\ & C & T & C & \\ & & \ddots & \ddots & \ddots \\ & & & C & T \end{bmatrix},$$

with

$$T = \begin{bmatrix} -20 & 4 & & & \\ 4 & -20 & 4 & & \\ & 4 & -20 & 4 & \\ & & \ddots & \ddots & \ddots \\ & & & 4 & -20 \end{bmatrix}, \quad C = \begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & 4 \end{bmatrix}.$$

Furthermore, the above matrix A can be constructed by using the **Kronecker product**

$$A = \frac{1}{6h^2}\left(S \otimes Q + Q \otimes S\right), \tag{8}$$

where

$$S = \begin{bmatrix} 10 & 1 & & & \\ 1 & 10 & 1 & & \\ & 1 & 10 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & 10 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & 1/2 & & & \\ 1/2 & -1 & 1/2 & & \\ & 1/2 & -1 & 1/2 & \\ & & \ddots & \ddots & \ddots \\ & & & 1/2 & -1 \end{bmatrix}.$$

Note that both matrices S and Q are **tridiagonal**.

- In MATLAB, we can construct A given by Eq. (8) via the following commands:

```
% m stands for the number of interior points
e = ones(m,1);
S = spdiags([e 10*e e], [-1 0 1], m, m);
Q = spdiags([1/2*e -e 1/2*e], [-1 0 1], m, m);
A = ( kron(Q,S) + kron(S,Q) ) / ( 6 * h^2 );
```