Please write legibly, explain clearly and present your solutions in an organised way. Some points will be given for the quality of presentation.

1. Consider the two-point (one-dimensional) fourth-order Boundary Value Problem (BVP)

$$u_{xxxx} + u = g \quad \text{in} \quad (a, b) \tag{1}$$

where u is periodic on (a, b). (This means that $u(a + x) = u(b + x)\ \forall x$.) Adjust the technique used in assignment 1, question 5, to handle the u-term in the differential operator, as well as the periodic boundary conditions. Note that, with the gridpoints being $x_i = i/n$, $i = 0, \cdots, n$, you will have n equations and n unknowns. (You can choose the equations/unknowns to correspond to points x_i, $i = 1, \cdots, n$, or to points x_i, $i = 0, \cdots, n-1$.) Let $A\bar{u} = \bar{g}$ be the linear system arising. (Note that $\bar{u}$ and $\bar{g}$ are vectors, while u and g are functions.)

(a) [*3 points*] Give A and $\bar{g}$, for $a = 0$, $b = 2\pi$, (i.e. the domain is $(0, 2\pi)$), $n = 6$, and assuming g is such that the solution to the BVP is $u = \sin x$. (Give numerical values for the entries of A, keeping the $1/h^4$ factor out. For $\bar{g}$, just give symbolic values, arising from g. Note that both A and $\bar{g}$ are simpler - and dull - compared to those of assignment 1.)

(b) [*17 points*] Adjust your programme for assignment 1, question 5(f), to discretize the above problem for various values of n, and to use the CG method to solve the linear system, with tolerance 10^{-8} and the zero vector as initial guess.
For $n = 8, 16, 32, 64, 128$, compute and output the maximum in absolute value error $e_n = \max_{i=1}^{n} |\bar{u}_i - u_i|$ of the approximation on the knots (assuming the exact solution u to the BVP is known). Also output the number of iterations required for the convergence of CG. Comment on the number of iterations for various values of n. Give a mathematical explanation of what you observe.
Hint: Find a set of real eigenvectors for A and compare with $\bar{g}$ and the exact solution $\bar{u}$ of the linear system.
Notes: You can use the `pcg` function in MATLAB as an implementation of the CG method, or your own implementation. If you use `pcg`, set the preconditioner to identity (i.e. `[]`).

2. [*10 points*] Let A be a $n \times n$ symmetric positive definite matrix, and assume that the Conjugate Gradient (CG) method is applied to $Ax = b$, for some $n \times 1$ vector b. Show that the direction vectors $d^{(k)}$, and the residual vectors $r^{(k)}$ generated by the CG method satisfy

$$r^{(k)} \in \text{span}(d^{(0)}, d^{(1)}, \cdots, d^{(k)}), \quad \text{for } k = 0, \cdots, n-1,$$
$$Ad^{(k)} \in \text{span}(d^{(0)}, d^{(1)}, \cdots, d^{(k+1)}), \text{ for } k = 0, \cdots, n-2,$$

and

$$\text{span}(d^{(0)}, d^{(1)}, \cdots, d^{(k)}) = \text{span}(d^{(0)}, Ad^{(0)}, \cdots, A^k d^{(0)})$$
$$= \text{span}(r^{(0)}, Ar^{(0)}, \cdots, A^k r^{(0)}), \text{ for } k = 0, \cdots, n-1.$$

3. [*10 points*] Let A be a $n \times n$ symmetric positive definite matrix, and assume that the CG method is applied to $Ax = b$, for some $n \times 1$ vector b. Show that the CG iterate $x^{(k+1)}$ minimizes $\|x - \hat{x}\|_{A^{1/2}}$ over all $\hat{x} \in x^{(0)} + \text{span}(d^{(0)}, \cdots, d^{(k)})$.

4. Consider the two-dimensional Boundary Value Problem (BVP)

$$au_{xxxx} + bu_{xxyy} + cu_{yyyy} + du_{xx} + eu_{yy} + fu = g \quad \text{in} \quad (0, 1) \times (0, 1)$$
$$u = \gamma \quad \text{on} \quad x = 0,\ x = 1,\ 0 \le y \le 1, \text{ and } y = 0,\ y = 1,\ 0 \le x \le 1$$
$$u_{xx} = \zeta \quad \text{on} \quad x = 0,\ x = 1,\ 0 \le y \le 1$$
$$u_y = \eta \quad \text{on} \quad y = 0,\ y = 1,\ 0 \le x \le 1$$

where $u(x, y)$ is unknown and $a(x, y)$, $b(x, y)$, $c(x, y)$, $d(x, y)$, $e(x, y)$, $f(x, y)$, $g(x, y)$, $\gamma(x, y)$, $\zeta(x, y)$ and $\eta(x, y)$ are given functions. Note that this is a generalization of the problem in assignment 2, question 3, with a modification in the boundary conditions along the horizontal boundary lines. Call this BVP "BVP I".

(a) [*10 points*] Consider the instance of the above BVP with $a = c = 1$, $b = 2$, and $d = e = f = 0$. Call this BVP "BVP II". Extend the techniques of assignment 2, question 3, to handle the new boundary conditions. Let P be the matrix arising from BVP II. Give a tensor product form of P using tridiagonal matrices, one pentadiagonal matrix and the identity. Give the block-diagonalization of P. Describe an FFT-based (FST) algorithm for solving $P\bar{u} = \bar{g}$, that applies

FST to one dimension and banded LU to the other. Note that this requires only a minor modification to the algorithm you gave in assignment 2.

(b) [*40 points*] Extend the techniques of assignment 2, question 3, to handle variable coefficients in the differential operator, the u_{xx}, u_{yy} and the u terms, and the new boundary conditions. Extend your programme so that it

(α) Generates the matrix A and the respective right-side vector $\bar{g}$, for the BVP I. The matrix should be generated and stored in sparse or banded format.

(β) Uses two linear solvers to solve $A\bar{u} = \bar{g}$ to obtain $\bar{u}_k$ on the (interior) gridpoints. The solvers are: (β1) Banded LU and (β2) Preconditioned GMRES (PGMRES), with preconditioner the matrix arising from BVP II, solved by the algorithm in part (a) of this question (i.e. FFT (FST) applied to one dimension and banded LU to the other). In MATLAB, you will need to pass the FFT solver of (a) as the preconditioning function argument of `gmres`.

(γ) Computes and outputs the maximum norm of the difference of the two solution vectors.

(δ) Outputs the number of iterations required for convergence of PGMRES.

(ε) Computes and outputs the maximum in absolute value error $\varepsilon_n = \max_{i=1,j=1}^{n,n} |\bar{u}_{ij} - u_{ij}|$ of the approximation on the knots (assuming the exact solution u to the BVP is known).

(στ) For pairs of values of n of the form $(n,\, 2n)$, computes and outputs the order of convergence, $\log_2(\varepsilon_n / \varepsilon_{2n})$, corresponding to $(\varepsilon_n,\, \varepsilon_{2n})$.

Let $a = 1 + \exp(x+y)$, $b = 1 + 1/(x+y)$, $c = 3 + xy$, $d = -\sin x \sin y$, $e = -1 - \exp(x+y)$, $f = x + y$, and g chosen so that the solution to the BVP I is $u = x^{9/2} y^{9/2}$.

Run your programme for $n = 8, 16, 32, 64, 128$. For `gmres` use $tol = 10^{-9}$, initial guess the zero vector, restart 20 (this will result in no restart), and a small maximum number of iterations. Make a table showing the errors, orders of convergence, number of iterations, and max norms of the difference between the two solution vectors. (Or format your output to produce the results in the form of a table.) Do **not** output more than requested. Compact your output so that it is easily read. (In MATLAB, use `format compact`, of the `fprintf` statement with appropriate format.)

Comment on how the number of iterations behaves with n, and how the residual behaves as the iterations proceed. You can make any other comments you find interesting.

Use double precision. (MATLAB uses double precision automatically.) Hand in a hard-copy of your code together with the output required, the table (if separate) and your comments.

Note: One could also use PCG (still with the same preconditioner) instead of PGMRES, However, due to unsymmetry, the convergence of PCG on the matrix in (b) is expected to be less robust than that of PGMRES.

(c) [*10 points*] Consider BVP II with $g = 1$, $\gamma = \zeta = \eta = 0$. The exact solution of this BVP is not known. You can solve the linear system arising from this problem by banded LU and by the solver in (a). Do this for $n = 8, 16, 32, 64, 128$, and output the approximate solutions at point (0.5, 0.5) of the domain obtained with the two solvers. What can you say about the order of convergence of the computed solutions? How big an n should we take in order to obtain the solution at (0.5, 0.5) with 10^{-9} precision?

Notes:

You are welcome to use the `kron`, `spdiags` and `speye` functions in MATLAB to build the matrix for this problem. Try to avoid, as much as possible, explicit for loops. Make sure that at no point in the programme you store a non-sparse matrix. You can use MATLAB's backslash (\) as a banded LU solver as long as you have your matrix generated as sparse. Find out *in advance* how the preconditioner is passed in MATLAB's `gmres` routine and ask me any questions you may have.