Please note that there are tow EXCEL files. One is the main file which has the calculations for 8 questions and the other one is Q4.1) which is used for the next 7 questions.

Q4.1)

In this question, Gaussian Elimination and Back substitution is the method which should be used. The objective is to convert the coefficient matrix to a diagonal matrix. So, the following is 10 steps to make the diagonal matrix. Please note that the results are verified using MMULT function:

30	-4	0	0	0	40
-4	30	-4	0	0	20
0	-4	30	-4	0	20
0	0	-4	30	-4	20
0	0	0	-4	30	40

1

30	-4	0	0	0	40
0	29.4667	-4	0	0	25.3333
0	-4	30	-4	0	20
0	0	-4	30	-4	20
0	0	0	-4	30	40

2

30	-4	0	0	0	40
0	29.4667	-4	0	0	25.3333
0	0	29.457	-4	0	23.4389
0	0	-4	30	-4	20
0	0	0	-4	30	40

3

30	-4	0	0	0	40
0	29.4667	-4	0	0	25.3333
0	0	29.457	-4	0	23.4389
0	0	0	29.4568	-4	23.1828
0	0	0	-4	30	40

4

30	-4	0	0	0	40
0	29.4667	-4	0	0	25.3333
0	0	29.457	-4	0	23.4389
0	0	0	29.4568	-4	23.1828
0	0	0	0	29.4568	43.148

5

1	-0.1333	0	0	0	1.33333
0	1	-0.1357	0	0	0.85973
0	0	1	-0.1358	0	0.7957
0	0	0	1	-0.1358	0.78701
0	0	0	0	1	1.46479

6

1	-0.1333	0	0	0	1.33333
0	1	-0.1357	0	0	0.85973
0	0	1	-0.1358	0	0.7957
0	0	0	1	0	0.98592
0	0	0	0	1	1.46479

7

1	-0.1333	0	0	0	1.33333
0	1	-0.1357	0	0	0.85973
0	0	1	0	0	0.92958
0	0	0	1	0	0.98592
0	0	0	0	1	1.46479

8

1	-0.1333	0	0	0	1.33333
0	1	0	0	0	0.98592
0	0	1	0	0	0.92958
0	0	0	1	0	0.98592
0	0	0	0	1	1.46479

9

1	0	0	0	0	1.46479
0	1	0	0	0	0.98592
0	0	1	0	0	0.92958
0	0	0	1	0	0.98592
0	0	0	0	1	1.46479

10

Double check

						MMULT
30	-4	0	0	0	1.46479	40
-4	30	-4	0	0	0.98592	20
0	-4	30	-4	0	0.92958	20
0	0	-4	30	-4	0.98592	20
0	0	0	-4	30	1.46479	40

In addition, no source was used since I knew how to do It.

$$x_1 = 1.46479, x_2 = 0.98592, x_3 = 0.92958, x_4 = 0.98592, x_5 = 1.46479$$

Q4.2)

LU decomposition is a method which also can be used to solve system of linear equation. The following is a summery of LU composition method:

$A = LU \text{ and substitute into } AX = B$

$sOLVE\ LUX = B \text{ for } X \text{ to solve the system.}$

$Let\ UX = Y$

$LY = B \text{ and } UX = Y$

$First\ Solve\ LY = B \text{ for } Y \text{ and then dolve } UX = Y \text{ for } X$

So, L and U matrices have been generated:

U

30	-4	0	0	0	40
0	29.46667	-4	0	0	25.33333
0	-4	30	-4	0	20
0	0	-4	30	-4	20
0	0	0	-4	30	40
1					

30	-4	0	0	0	40
0	29.46667	-4	0	0	25.33333
0	0	29.45701	-4	0	23.43891
0	0	-4	30	-4	20
0	0	0	-4	30	40
2					

30	-4	0	0	0	40
0	29.46667	-4	0	0	25.33333
0	0	29.45701	-4	0	23.43891
0	0	0	29.45684	-4	23.1828
0	0	0	-4	30	40
3					

30	-4	0	0	0	40
0	29.46667	-4	0	0	25.33333
0	0	29.45701	-4	0	23.43891
0	0	0	29.45684	-4	23.1828
0	0	0	0	29.45683	43.14804
4					

L

1	0	0	0	0
-0.13333	1	0	0	0
0	-0.13575	1	0	0
0	0	-0.13579	1	0
0	0	0	-0.13579	1

L and U matrices were verified and the original coefficient matrix was resulted. Next, Y and X matrices are calculated:

Y Matrix						
1	0	0	0	0	40	40
-0.13333	1	0	0	0	25.33333	20
0	-0.13575	1	0	0	23.43891	20
0	0	-0.13579	1	0	23.1828	20
0	0	0	-0.13579	1	43.14804	40

X Matrix						
30	-4	0	0	0	1.464789	40
0	29.46667	-4	0	0	0.985915	25.33333
0	0	29.45701	-4	0	0.929577	23.43891
0	0	0	29.45684	-4	0.985915	23.1828
0	0	0	0	29.45683	1.464789	43.14804

The results are very similar to what calculated in Q4,1

$(x_1 = 1.46479, x_2 = 0.98592, x_3 = 0.92958, x_4 = 0.98592, x_5 = 1.46479)$

Source: https://www.youtube.com/watch?v=UlWcofkUDDU

Q4.3)

In this question, Cholesky decomposition is the method which has been used. Here is a summary for this method:

$$A = LL^T$$

Where L is a lower triangular matrix and L^T is the conjugate transpose of L.

LT				
5.477226	-0.7303	0	0	0
0	5.428321	-0.73688	0	0
0	**0**	5.427432	-0.737	0
0	**0**	**0**	5.427415	-0.737
0	**0**	**0**	**0**	5.427415
L				
5.477226	**0**	**0**	**0**	**0**
-0.7303	5.428321	**0**	**0**	**0**
0	-0.73688	5.427432	**0**	**0**
0	0	-0.737	5.427415	**0**
0	0	0	-0.737	5.427415

This is calculated which has verified by $A = LL^T$

The similar to LU decomposition method, Y and X matrices are calculated:

Matrix Y

5.4772256	0	0	0	0	7.302967	40
-0.7302967	5.428321	0	0	0	4.666882	20
0	-0.73688	5.427432	0	0	4.318601	20
0	0	-0.737	5.427415	0	4.271425	20
0	0	0	-0.737	5.427415	7.950016	40

Matrix X

5.4772256	-0.7303	0	0	0	1.464789	7.302967
0	5.428321	-0.73688	0	0	0.985915	4.666882
0	0	5.427432	-0.737	0	0.929577	4.318601
0	0	0	5.427415	-0.737	0.985915	4.271425
0	0	0	0	5.427415	1.464789	7.950016

The results are correct!

As you can see, the obtained results are correct!

Source: https://en.wikipedia.org/wiki/Cholesky_decomposition

Q4.4)

Gauss-Jordan method is used in this question which is summarized as:

$$[A|I] \Rightarrow [I|B]$$

Inversion matrix was calculated through 9 steps (please go to excel file)

30	-4	0	0	0	1	0	0	0	0
-4	30	-4	0	0	0	1	0	0	0
0	-4	30	-4	0	0	0	1	0	0
0	0	-4	30	-4	0	0	0	1	0
0	0	0	-4	30	0	0	0	0	1

1	0	0	0	0	0.03395	0.00461	0.00063	0.00008	0.00001
0	1	0	0	0	0.00461	0.03458	0.00470	0.00064	0.00008
0	0	1	0	0	0.00063	0.00469	0.03459	0.00469	0.00063
0	0	0	1	0	0.00008	0.00064	0.00469	0.03457	0.00461
0	0	0	0	1	0.00001	0.00008	0.00063	0.00461	0.03395

The inversion matrix was used to calculate the X-matrix and results were correctly calculated.

Q4.5)

Jacobi Iterative method was used which is summarized as:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

While x_j^k is the parameters from previous iteration. The following figure is iterations of Jacobi Iterative method:

x1	1	First Iteration	x1	1.466667	Second Iteratio
x2	1		x2	0.933333	
x3	1		x3	0.933333	
x4	1		x4	0.933333	
x5	1		x5	1.466667	
x1	1.457778	Third Iteration	x1	1.464889	Fourth Iteration
x2	0.986667		x2	0.983111	
x3	0.915556		x3	0.929778	
x4	0.986667		x4	0.983111	
x5	1.457778		x5	1.464889	
x1	1.464415	Fifth Iteration	x1	1.464794	Sixth Iteration
x2	0.985956		x2	0.985766	
x3	0.92883		x3	0.929588	
x4	0.985956		x4	0.985766	
x5	1.464415		x5	1.464794	
		1.464789	Answer		
		0.985915			
		0.929577			
		0.985915			
		1.464789			

Source: https://en.wikipedia.org/wiki/Jacobi_method

Q,4.6)

This method is also very similar to Jacobi Iterative method. The inly difference is that, it uses updated value for those parameters which have already been recalculated in the same iteration.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

Slight modification was applied to the Jacboi Iterative method for this method. The results show better accuracy:

x1	1	First Iteration	x1	1.466667	Second Iteration
x2	1		x2	0.995556	
x3	1		x3	0.932741	
x4	1		x4	0.924365	
x5	1		x5	1.456582	
x1	1.466074	Third Iteration	x1	1.464868	Fourth Iteration
x2	0.986509		x2	0.984842	
x3	0.92145		x3	0.929144	
x4	0.983738		x4	0.985819	
x5	1.464498		x5	1.464776	
x1	1.464646	Fifth Iteration	x1	1.464778	Sixth Iteration
x2	0.985839		x2	0.985911	
x3	0.929554		x3	0.929576	
x4	0.985911		x4	0.985915	
x5	1.464788		x5	1.464789	
		1.464789	Answer		
		0.985915			
		0.929577			
		0.985915			
		1.464789			

The results were compared with results obtained in Q4.1).

Source: https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method

Q,4.7)

In this method, ω term was added to the formula of Q,4.6 in this form:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

And all calculations are dependent on the term ω.

It was concluded that $\omega = 1.02$ results in more accuracy. In this report, the last two iterations are shown:

x1	1.464629		
x2	0.985922		
x3	0.929578		
x4	0.985916	W	1.02
x5	1.464789	Answer	Error
x1	1.464793	1.464789	4.13933E-06
x2	0.985916	0.985915	5.04399E-07
x3	0.929578	0.929577	6.27667E-08
x4	0.985916	0.985915	7.99754E-09
x5	1.464789	1.464789	1.03823E-09
		Sum	4.71553E-06

Source: https://en.wikipedia.org/wiki/Successive_over-relaxation

Q,4.8)

The following formula is for Thomas algorithm:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n-1 \end{cases}$$

and

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n. \end{cases}$$

The solution is then obtained by back substitution:

$$x_n = d'_n$$
$$x_i = d'_i - c'_i x_{i+1} \qquad ; \ i = n-1, n-2, \ldots, 1.$$

For:

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

The formulas were applied for our case and the following was resulted:

30	-4	0	0	0	40
-4	30	-4	0	0	20
0	-4	30	-4	0	20
0	0	-4	30	-4	20
0	0	0	-4	30	40
i	Ci'	di'			
1	-0.13333	1.333333			
2	-0.13575	0.859729			
3	-0.13579	0.795699			
4	-0.13579	0.787009			
5		1.464789			
x1	1.464789	1.464789			
x2	0.985915	0.985915			
x3	0.929577	0.929577			
x4	0.985915	0.985915			
x5	1.464789	1.464789			
	Correct!				

Source: https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm